Dino Game AI

Tarmo Kullas ja Mikko Maran

Kood: https://github.com/tarmokullas/TI-dinogame

Töö eesmärk ja tehisintellekti kasutamise vajadus

Töö eesmärgiks oli luua Google tuntud dinosauruse mäng ning treenida tehisintellekt, mis oskaks seda mängu mängida.

Mängu kirjeldus: eesmärgiks on mitte tabada erinevaid takistusi, mis mängija ehk dinosauruse teele tulevad. Takistusteks võivad olla erinevate suurustega kaktused, millest tuleb üle hüpata, või linnud, mille puhul tuleb kummardada. Mängu edenedes suureneb kiirus, millega takistused mängijale lähenevad.

Originaali on võimalik mängida siin: https://elgoog.im/t-rex/

Kasutatud ideed (nii kursuse materjalidest kui mujalt) koos viidetega

https://youtu.be/sB_IGstiWlc – ülesande üldine idee, ennustamiseks vajalikud sisendid

https://youtube.com/playlist?list=PLzMcBGfZo4-lwGZWXz5Qgta_YNX3_vLS2 – ülesande täitmiseks sobilik tehisintellekti idee, üldine programmi ülesehitus

https://neat-python.readthedocs.io/en/latest/config_file.html - TI algoritm ja selle seadistused

https://stackoverflow.com/questions/61365668/applying-saved-neat-python-genome-to-test-environment-after-training - salvestatud genoomi salvestamine ning laadimine ja selle mängima panemine

https://github.com/codewmax/ChromeDinosaur - mängu visualid ja loogika

Töö autori enda panuse kirjeldus

Tarmo: mängu ja selle loogika loomine, tehisintellekti implementeerimine ja testimine

Mikko: mängu visuaalid, parima genoomi ja tulemuse salvestamine, mängumenüü ja selle valikute (mängi ise, parim genoom ja treeni nullist) implementeerimine

Programmi testimisvõimalused

Programmi on võimalik kasutada kolmel viisil:

a) ise mängu mängides
b) lasta tehisintellektil mäng ise selgeks õppida

c) lasta tehisintellektil mängida varasemalt juba treenitud algoritmiga

Mainiksime ära, et tehisintellektil on inimese ees siiski eelis. TI näeb infot järgmise takistuse kohta isegi siis, kui takistus pole veel ekraanile ilmunud. Inimene saab selle info alles takistuse ekraanile ilmumisel.

Töö käigu kirjeldus: millised olid probleemid, mis õnnestus, mis jäi realiseerimata jne (kirjeldatud probleemid ega puudused ei mõju hindele kuidagi negatiivselt)

Esimene probleem oli Pygame tööle saamine Google Colabis. Tahtsime kindlasti Pygame'i kasutada, kuna tundsime end selles kindlalt. Siiski ei õnnestunud meil leida head juhist, kuidas seda Colabis tööle saada. Et mitte sellele liigselt aega raisata otsustasime ülesande teha klassikalise pythoni programmina.

Järgmisena oli murekohaks, et kas kirjutada mängu enda kood ise või kasutada valmisolevat, et panna rohkem rõhku tehisintellekti poolele, kuna tundides me ei olnud geneetilist TI algoritmi õppinud. Kuigi alustasime esimese variandiga, kus kasutasime juba loodud mängu, siis poole peal loobusime sellest ja kirjutasime ise oma, et saaksime seda oma TI algoritmiga lihtsamini integreerida.

Üsna suureks probleemiks oli ka TI implementeerimine, kuna meil polnud täpset arusaama, kuidas seda geneetilise algoritmi puhul teha tuleks. Lõpuks kasutasime NEAT-Pythonit, mis võrreldes kõikide teiste leitud lahendustega oli meie jaoks kõige arusaadavam. Siiski läks ka selle algoritmiga aega. Võrreldes juhistega (Flappy Bird-i mäng) oli meie omal rohkem väljundeid. Flappy Birdis oli üks väljund hüppa, meil aga hüppa, kummarda, jookse (ehk ei hüppa ega kummarda). Alguses me ei taibanud täpselt, kuidas peaksime õige tulemuse kätte saama. Rohke testimine ja katsetamine aitas. Veel oli probleem selles, et meie Dino ei tahtnud linnu puhul kummardamist ära õppida ning proovis alati sellest üle hüpata, mis võrdus Dino surmaga. Selle saime korda, kui lisasime fitness funktsiooni preemiad ja karistused erinevatele käitumistele linnu puhul. Näide 1: iga kokkupõrge andis -100 fitnessi, aga kui see toimus kummardusest liiga vara püstitõusmisel, mis puhul linnuga kokku põrgati, siis andsime 20 fitnessi punkti tagasi. Näide 2: kui järgmine takistus oli lind ja jooksid selle poole kummardades sai Dino samuti fitnessi juurde.

Üks lahendamata probleem on TI eelis inimese ees, mida eelnevalt mainisime. Kuna avastasime selle töö lõpuosas, siis ei soovinud me enam koodi selle pärast ekstra muutma hakata. Üheks ja tõenäoliselt ausamaks võimaluseks oleks sobinud, kui oleksime TI-le andnud info järgmise takistuse kohta alles siis, kui see ekraanile ilmub. Teine variant oleks olnud järgmise takistuse info kuvamine inimesele, kuid tõenäoliselt oleks suuremate mängukiiruste puhul ikkagi arvutil eelis tekkinud, kuna arvuti suudab neid numbreid kiiremini/paremini protsessida kui inimene.

Parem võiks olla ka Dino ja takistuse kokkupõrke määramine. Hetkel on kontrollitakse, kas kujutiste ümber olevad kastid puutuvad omavahel kokku, kuid see võiks olla täpsem ehk kontrollitakse, kas reaalsed pikslid omavahel puutuvad kokku.

Tulemused

Lõplikku geneetilist algoritmi testisime 100-se populatsiooniga ning 10 generatsiooni vältel. Selle ajaga suutis parim genoom õppida mängu nii hästi selgeks, et lõplik skoor tuli 45000. Võrdluseks võib öelda, et meie mõlema parimad skoorid ise mängides jäid 4000 ringi, seega tehisintellekt saavutas juba vaid 10 generatsiooniga meist 10 korda parema skoori. Ent veelkord tuletame meelde, et tehisintellektil on inimese ees siiski eelis, kuna TI näeb takistust isegi siis, kui see pole veel ekraanil.
Kõrvalmärkusena tuleb mainida, et see genoom vahepeal ei saa hakkama kõige suuremast kaktusest üle hüppamisega kõige madalamal mängu kiirusel, ehk see genoom ei ole siiski veel nii tark. Kindlasti tuleks rohkemate generatsioonidega treenimisel skoor kõrgem ja tehisintellekt targem. Sedasama genoomi ongi võimalik praegu panna mängima, kui valida menüüst “Run all-time best genome”.

Kokkuvõte

Projekti võib lõppkokkuvõttes õnnestunuks lugeda, kuna me ei olnud varem geneetilise algoritmiga kokku puutunud, kuid lõpuks saime hakkama sellise tehisintellekti loomisega, kes õppis mängu meist palju paremini ära. Kui lõpuks nägime 100 dinosaurust hüppamas ja et need ka järjest paremaks läksid, siis tuli hea meel küll.