16. GitHub Copilotin ongelmanratkaisukykyjen analysointi

Esitiedot:

Github Copilot on Microsoftin, OpenAi:n ja GitHubin kehittämä avoimeen lähdekoodiin perustuva ohjelmointiavustin. Sen pystyy lataamaan omalle ohjelmointi alustalle, ja minä tässä työssä latasin sen Visual Studio Codeen. GitHub Copilot on maksullinen lisäosa. Laajaa taustatietietoa kyseisestä tekoälymoduulista minulta löytyy vähänaika sitten palautetulla kandidaatin työstäni aiheena ”GitHub Copilot -tekoälymoduulin käyttö ohjelmoinnin tukena”

Aiempia havaintoja:

GitHub Copilotilla tehtyjen empiiristen tutkimusten mukaan kyseisellä botilla on taipumusta noin (vähän alle) puolet vasteista generoida virheellistä ja tietoturvariskejä sisältävää koodia. Joissakin tutkimuksissa GitHub Copilot on kuitenkin osannut helppoihin algoritmeihin antaa oikean vastauksen paremmin kuin nuoremmat ohjelmoijat. Omien aiempien empiiristen tutkimusten mukaan Pilotti antoi omaan web porjektiini ihan hyviä vasteita, kömpleyyttä esiintyi hiukan suomen ja englannin kielen vaihtelevuudessa.

Tavoite:

Vertaillaan muutaman globaalisti tunnetun yksinkertaisen pelin muodostamista (matopeli, tetris, ristinolla). Annetaan GitHub tehdä vasteet Pythonilla ja Javalla. Analysoidaan vasteen potentiaalia ja, sitä kuinka paljon pitää itse muuttaa koodia tai antaa lisätietoja, jotta Copilotin avustuksella saadaan toimiva/ajettava ratkaisu. Myös analysoidaan ratkaisun ”kömpelyyttä”.

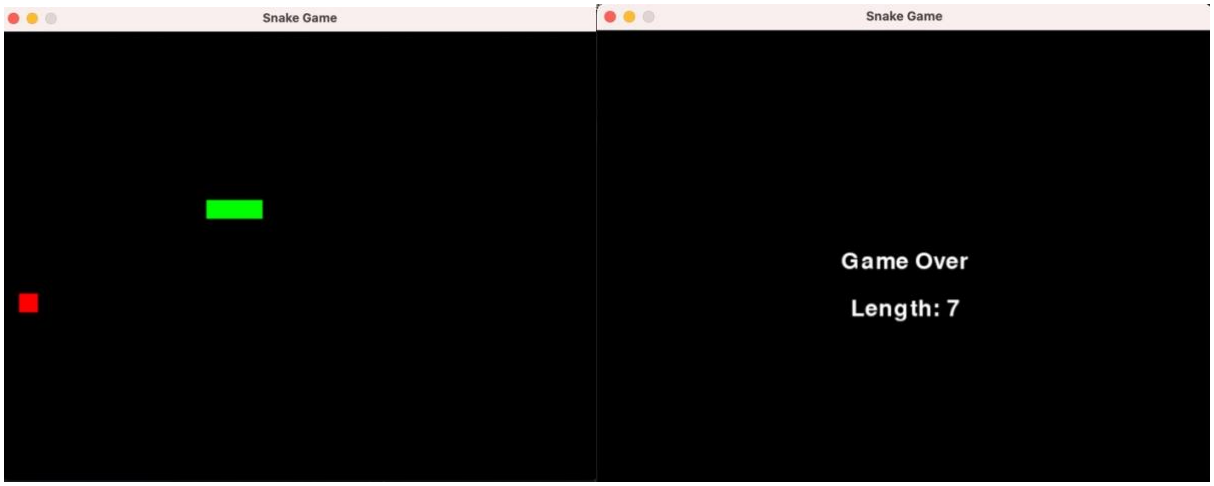
Projekti:

Matopeli:

PYTHON:

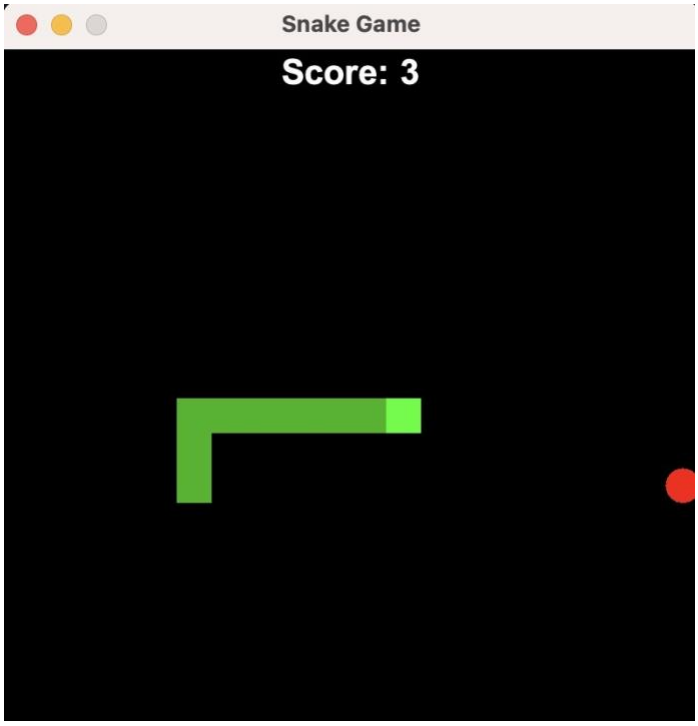
Aloitin kokeilemalla, miten GitHub Copilot pystyy ohjelmoimaan matopelin Pythonilla käyttäen hakulausekkeena "generate snake game". Sain yllättävän hyvän, mutta ei täydellisen tuloksen. Copilot hyödynsi pygame-kirjastoa pelin grafiikan luomiseen. Matoa (vihreä) ohjattiin nuolinäppäimillä, ja mato kasvoi syödessään herkkuja, jotka ilmestyivät kentälle.

1. Versio: ”Generate snake game”
2. Versio: Can you change it so that the snake comes back into the game from the same spot on the opposite side?"
3. Versio: ”Modify it so that when the snake eats itself, a 'game over' appears on the screen and the length of the snake at the end of the game”



JAVA:

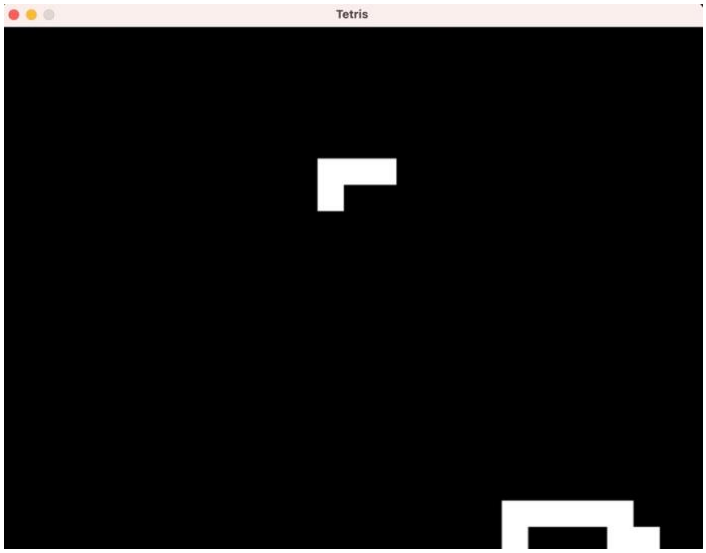
Aloitin generoinnin taas lauseilla: ”generate snakegame”. Java-versio matopelistä oli selvästi edistyneempi kuin Pythonilla toteutettu. Pisteet näkyivät jatkuvasti ruudulla, ja madon päästä häntään kulki hienovarainen häivytys. Lisäksi omena oli muotoiltu pyöreäksi. Aloin pohtia, kuoleeko mato yleensä seiniin, sillä tässä versiossa näin tapahtui. Toisin kuin Pythonin ensimmäisessä versiossa, Javassa peli ilmoitti "game over" -viestillä, kun mato kuoli. Java antoi siis ensimmäisellä kerralla toimivan ja ajettavan vasteen.



Tetris:

PYTHON:

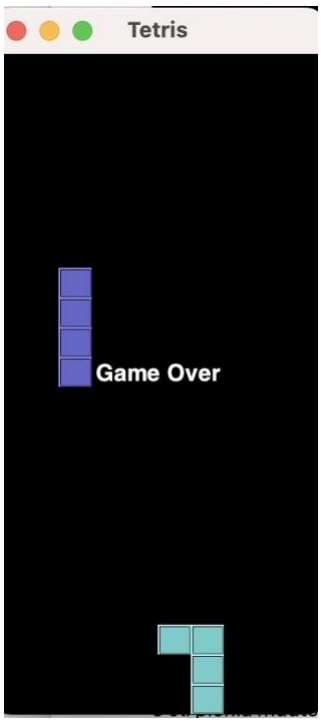
Aloitin käyttämällä hakusanaa "generate tetris". Tällä kertaa Copilot loi pelkän perustan Tetrikselle. Se loi pygame-ikkunan, mutta käyttäjän syötteiden määrittely ja pelilogiikka puuttuivat, joten ainoana tuloksena oli musta ruutu. Toisessa versiossa pyysin Copilotia luomaan pelattavan version, mikä lisäsi koodin määrää. Kun koodin ajoi, se pysyi näkyvissä vain noin sekunnin ajan ennen kaatumista, mutta ehdin nähdä Tetrikselle tyypillisen muodon näytön yläreunassa. Kolmannessa versiossa aloin tarkemmin tutkia koodin virheitä ja pyysin Copilotilta useita kertoja parannuksia. Lopulta sain jokseenkin toimivan version ja sitä ainakin jokseenkin pystyi pelamaan.



JAVA:

Ensimmäiset kolme yritystä eivät tuottaneet toivottua tulosta. Neljännellä yrityksellä aloin käydä läpi havaitsemiani ongelmia yksi kerrallaan Copilotin kanssa. Huomasin esimerkiksi, että jotkin metodit olivat määrittelemättä. Menin suoraan kyseisen metodin kohdalle ja ohjeistin, että se täytyy määritellä. Viidennessä yrityksessä tehtiin pieniä muutoksia. Kuudennessa yrityksessä tarkastelimme ja korjasimme ongelmia jälleen yksitellen. Tämän seurauksena koodi alkoi toimia paremmin: pelissä palikat tippuvat kuten Tetriksessä, mutta taustalla näkyy vielä "game over" -teksti. Käyttöliittymä on huomattavasti hienostuneempi kuin Python-versiossa. Peli toimii nyt siten, että täysi rivi palikoita poistuu pelialueelta.

Seitsämännessä yrityksessä Copilotin ja minun yhteistoimin saimme aika hyvin toimivan sekä pelattavan version. Loppujen lopuksi Tetris -peliin Copilot tuotti noin 500 rivisen koodin.



RISTINOLLA:

Python

Python-koodin kehitys tictactoe-pelille alkoi yksinkertaisella versiolla, jossa käyttäjä syöttää haluamansa ruudun numeron terminaalin kautta. Tämä versio oli perusmuodoltaan toimiva, mutta kaatui, jos pelaaja yritti laittaa nappulan jo varatulle paikalle. Toisessa versiossa pyysin Copilotia korjaamaan ongelman, jossa peli kaatui, jos käyttäjä syötti virheellisen numeron. Copilot paransi peliä siten, että jos pelaaja yritti asettaa nappulan jo varattuun ruutuun, terminaali ilmoitti 'invalid number, try again', eikä peli kaatunut. Kuitenkin, jos syötettiin numero, joka ei vastannut yhtäkään pelialustan ruutua, peli kaatui edelleen. Kolmannessa versiossa pyysin vielä kerran korjaamaan tämän viimeisen virheen. Vaikka Copilot aluksi kirjoitti koko koodin uusiksi, pyysin sitä rajaamaan syötteet pelialustan sallimiin numeroihin. Tämän

jälkeen Copilot onnistui luomaan koodin, joka estää peliä kaatumasta väärillä syötteillä, ja peli toimi moitteettomasti.

JAVA:

JAVA:

Alun perin ristinollapeli ei toiminut, koska se sijaitsi omassa 'ristinolla'-nimisessä kansiossani. Ongelman ratkaisin lisäämällä manuaalisesti koodiin package ristinolla -määrittelyn, jonka jälkeen peli alkoi toimia. Peli pyysi syöttämään rivien ja sarakkeiden määrän erikseen, mutta ensimmäisellä yrittämällä peli kaatui, kun syötin rivien ja sarakkeiden määrän yhtä aikaa. Korjattuani syöttötavan, peli ei kaatunut, vaikka syötin tahallisesti olemattomia ruutukoordinaatteja kuten 5, 5; sen sijaan se vain pyysi pelaajaa yrittämään uudelleen ilman virheilmoitusta.

Myöhemmin pyysin Copilottia rajoittamaan sallittuja syötteitä tarkemmin. Vaikka aluksi se vain paransi syötteiden ohjeistusta, jatkokehityksessä Copilot onnistui lisäämään koodiin virheilmoituksen toiminnon, joka aktivoituu minkä tahansa virheellisen syötteen kohdalla. Nyt koodi toimii virheettömästi ja antaa johdonmukaisesti virheilmoituksen 'invalid number, try again' väärillä syötteillä.

Yhteenveto:

	JAVA	**JAVA**	**PYTHON**	**PYTHON**
Skenaario	Yritykset/Ajettava	Yritykset/Toimiva	Yritykset/Ajettava	Yritykset/Toimiva
Matopeli	1	1	1	3
Ristinolla	1	2	1	3
Tetris	4	7+	3	3(+)

(Taulukkoon laitettu yritykset siihen, että koodi oli ajettava, joka tarkoitti, että runnauksen jälkeen tapahtui jotain. Pelkkä valkoinen/musta ruutu ei tosin laskettu siihen, että se olisi ajettava. Toimivaksi laskin ne vasteet, joissa peli oli sääntöjen mukaan pelattavissa eikä sen testauksen mukaan kaatuillut.)

Päätelmiä:

Johtopäätöksenä voidaan todeta, että GitHub Copilot tuottaa yksinkertaisissa skenaarioissa lähes välittömästi toimivaa koodia. Vertailuna, fuksivuotenani kehittämäni Pythonilla toteutettu ristinollapeli vaati useamman päivän pohdintaa toimiakseen, sillä tuolloin koodauskokemukseni oli vähäistä. Copilotin avulla voi kuitenkin nopeasti ja pienellä vaivalla saada aikaan laadukasta koodia, mikäli käyttäjällä on perustiedot koodauksesta. Tämä mahdollistaa tarkemmat ja kohdennetummat ohjeet koodin ongelma-alueille. Vaikka ohjeistaminen on mahdollista ilman syvää koodausosaamista, tällöin on riski, että Copilot ei ymmärrä ohjeistusta oikein, mikä voi johtaa väärinymmärryksiin ja tehottomaan koodikorjaukseen. Riippuen haluttavan koodin vaikeustasosta, koodikielestä sekä hakulauseiden oikeasta käytöstä, voidaan saada erittäin pätevää ja hyvää ohjelmoinnin apua GitHub Copilotin avulla.