Groundhog-projektin 2. koodikatselmuksen kirjalliset huomiot

Juuso Tuononen

2.6.2021

1 Yleistä

Koodikatselmuksessa on käytetty koodeja commitista `cf91a5680d3a7b3d449361a077cac2eba0fd8145`.

Koodikatselmuksessa käytetty tietokone on edelleen sama kuin 1. koodikatselmuksessa käytetty tietokone:

- Intel Core i5-3470
- RAM 16 GB
- NVIDIA GeForce GTX 1050 Ti
- Samsung SSD 860 EVO 1TB
- Windows 10 Pro 20H2

2 `CoreLibrary`

2.1 `FileManager.cs`

ReadFileText-funktion dokumentaatioon olisi hyvä kirjoittaa mikä on callback-funktion bool-tyyppisen parametrin tarkoitus.

Thread kannattanee vaihtaa Taskiksi, koska Task on korkeamman tason abstraktio. Käytännössä kannattanee tehdä funktio suoraan C#:n async-ominaisuudella.
Lisätietoja: `https://stackoverflow.com/questions/13429129/task-vs-thread-differences`

2.2 `MapReader.cs`

NaturalSort-funktion Int32.TryParse-funktiokutsun onnistumista ei tarkisteta. Onko kyseessä ominaisuus?

ReadImgSeq-funktion yläpuolella on `#nullable enable` ja funktion jälkeen on `#nullable disable`. Ehkä kannattaa asettaa null-tarkistukset päälle koko lähdekoodille? Lisätietoja: `https://docs.microsoft.com/en-us/dotnet/csharp/nullable-references#nullable-contexts`

RawSlices-funktiossa rivillä 223 kertolaskun `width * height * byteDepth` tulos voi pyörähtää ympäri. Kannattanee tarkistaa onko pyörähdystä tapahtunut. `https://stackoverflow.com/questions/2056445/no-overflow-exception-for-int-in-c` Mikäli kyseessä olisi yhteenlasku, niin korjaus `(long) width + height + byteDepth` toimisi.

Virheenkäsittely näyttää olevan vielä enimmäkseen toteuttamatta. Esimerkiksi poikkeus MagickException on käsittelemättä. Lisäksi virheistä ei palaudu lisätietoja funktion kutsujalle.

MakeRawYStack-funktio ja rivillä 389 oleva kolmen luvun kertolasku. Mahtuuko `int * int * int` 64-bittiseen lukuun?

Edellisessä funktiossa ja rivillä 395 on `fs.Close()`. Using-lause varmaankin kutsuu Close-metodia automaattisesti?

Kuvia tallentavissa funktioissa MakeRawXStack, MakeRawYStack, MakeSeqXStack, MakeSeqYStack näyttää olevan jonkin verran copy-paste koodia.

Kuvien leveys ja korkeus kannattaisi ehkä ilmaista etumerkittömillä numeroilla, koska ne eivät ole negatiivisia koskaan.

2.3 `VectorManager.cs`

Dokumentaatiota ei ole vielä kirjoitettu tänne ja osa funktioista on sellaisia, että niiden toiminta ei ole ensisilmäyksellä selkeää, joten dokumentaation lisääminen tänne olisi hyvä juttu.

2.4 `Map.cs`

Seuraava on lähinnä refaktorointi-idea jatkokehitykseen. Map-luokassa ByteDepth-attribuutti näyttää olevan käytössä vain raw-kuville, niin ehkä tiedostoformaattiin liittyvät asiat kannattaisi erottaa jotenkin Map-luokasta?

2.5 Sample.cs

Metodeissa GetSliceX ja GetSliceZ tavutaulukossa olevan datan formaatti olisi hyvä olla dokumentoituna.

3 GroundhogApp

3.1 ConnectorDialog.xaml.cs

Luokan ConnectorDialog konstruktorissa on kommentti Coordinates-attribuutin päivitykseen liittyen. Onko kyseiseen attribuuttiin liittyvä WPF:n datan sidonta asetettu olemaan kaksisuuntainen? `https://docs.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-5.0#direction-of-the-data-flow`

Metodissa OpenBtn_Click ohjelma avaa ikkunan, mistä käyttäjä valitsee tiedoston. Mikäli tiedoston nimi vaihtuu juuri sen jälkeen, kun kyseinen ikkuna sulkeutuu, niin ohjelma kaatuu poikkeukseen System.IO.FileNotFoundException.

3.2 OpenMapDialog.xaml.cs

Metodissa CalculateRawRequestedSize on kertolasku int-luvuilla, jotka ovat muutettu ulong-tyyppisiksi. Mahtuuko kertolaskun tulos aina 64-bit lukuun?

3.3 MapControl.xaml.cs

Luokan MapControl konstruktorissa sijaitseva rivi 50 on kommentoitu. Tämänkaltaiset kommentit kannattaa siivota pois koodista tai kommentoida miksi kyseinen koodi on kommentoitu.

Metodi ZoomBox_TextChanged ja rivi 614. Value-muuttuja on tyyppiä uint, joten se ei mahdu 32-bittiseen float-lukuun. Kannattaa käyttää doublea.

Metodi ZoomSlider_ValueChanged ja rivi 633. Muuttuja val on float-tyyppiä. Liukuluvuissa kannattaa suosia yleensä doublea (64-bittinen), koska se on tarkempi verrattuna floattiin (32-bittinen).

3.4 MainWindow.xaml.cs

Metodeissa CreateXStack_Click ja CreateYStack_Click kuvapino tallennetaan asynkronisesti. Käyttälle ei tällä hetkellä kerrota tallennuksesta mitään. Erityisesti tällöin on vaarana, että ohjelma suljetaan kesken tallennuksen, josta mahdollisesti aiheutuu rikkinäisiä kuvatiedostoja.

3.5 `MainWindowViewModel.cs`

Onkohan metodeissa SaveSample ja SaveAsSample sellainen riski, että `this.sample` voi muuttua kesken tallennuksen? Lisäksi käyttäjälle ei ilmoiteta onko tallennus valmis. Ohjelman voi varmaankin sulkea kesken tallentamisen. Mitä tapahtuu jos SaveSample tallentaa ja samaan aikaan aletaan tallentamaan SaveAsSamplella samaan tiedostoon mihin SaveSample on tallentamassa? Samplen tietojen muuttumiseen liittyvät ongelmat korjaantuvat serialisoimalla sample synkronisesti ennen await-operaattorin käyttämistä koodissa.

Metodit CreateXStack ja CreateYStack ovat async-metodeja, joten niiden sisältö pyörii mahdollisesti eri säikeessä kuin missä `this.sample` sijaitsee. Täten sample mahdollisesti muuttuu kahdessa säikeessä ja tästä aiheutuu mahdollisesti sample-olion dataan liittyviä kilpa-ajotilanteita. Ratkaisuna on tehdä kyseisistä metodeista funktioita, joihin annetaan argumenttina kopiot tarvittavista tiedoista tai muuttaa MapReader.MakeSeqXStack ja muut kuvapinon tallentamiseen liittyvät funktiot async-funktioiksi.

Eli yhteenvetona vielä, että käytännössä molemmissa JSON-tiedoston ja kuvapinon tallentamisessa on sama säieturvallisuusongelma. Tässä pitäisi huolehtia, että data pysyy koko ajan muuttumattomana silloin, kun se on käytössä mahdollisesti useassa eri säikeessä.

3.6 `Utilities.cs`

Funktio PrepareHeader ja rivi 263. Tuohon kannattaa kommentoida mikä merkki `\x2024` on.

4 Muuta

Visual Studio näyttää 14 varoitusta ja 138 viestiä koodiin liittyen. Kyseiset huomiot kannattaa käydä läpi ja mikäli huomio vaikuttaa hyvältä, niin kannattaa tehdä niihin liittyvät muokkaukset koodiin. Yleisesti ottaen ohjelmoidessa kannattaa ottaa käyttöön ja hyödyntää lähdekoodin laatua tarkkailevia työkaluja (englanniksi linters). `https://en.wikipedia.org/wiki/Lint_(software)`

Ohjelma kaatuu, jos raw-tiedoston avaa 3D Tiff -tiedostona. Poikkeus ImageMagick.MagickCoderErrorException on käsittelemättä tiedoston MapReader.cs rivillä 246. Kyseinen koodirivi on funktiossa ThreeDTifSlices.

5 Yhteenveto

Koodin luettavuus on hyvää ja sitä oli mukava lukea. Lisäksi hyvä juttu oli, että dokumentaatiokommentteja on kirjoitettu kehityksen aikana. Mikäli dokumentoi kehityksen aikana, niin asiat ovat tuoreessa muistissa ja varmaankin joitain oleellisia yksityiskohtia muistaa dokumentoida paremmin verrattuna siihen, että lisää dokumentaatiokommentit viimeisenä tehtävänä.

Sanoisin, että suurin ongelma koodin kanssa on puutteet ohjelman virheenkäsittelyssä. Monet funktiot ja metodit voivat heittää virhetilanteen myötä poikkeuksen, joka käsittelemättömänä kaataa ohjelman. Toinen maininnan arvoinen ongelma on säieturvallisuus, joka lienee edellistä helpommin korjattava ongelma.

Kohdassa 2.2 mainittu null-tarkistusten asettaminen päälle koko lähdekoodille olisi varmaan ihan hyvä muutos. Null-viitteissä on omat ongelmansa. Kannattaa toki miettiä, että mitä korjauksia tai hienosäätöä priorisoi.