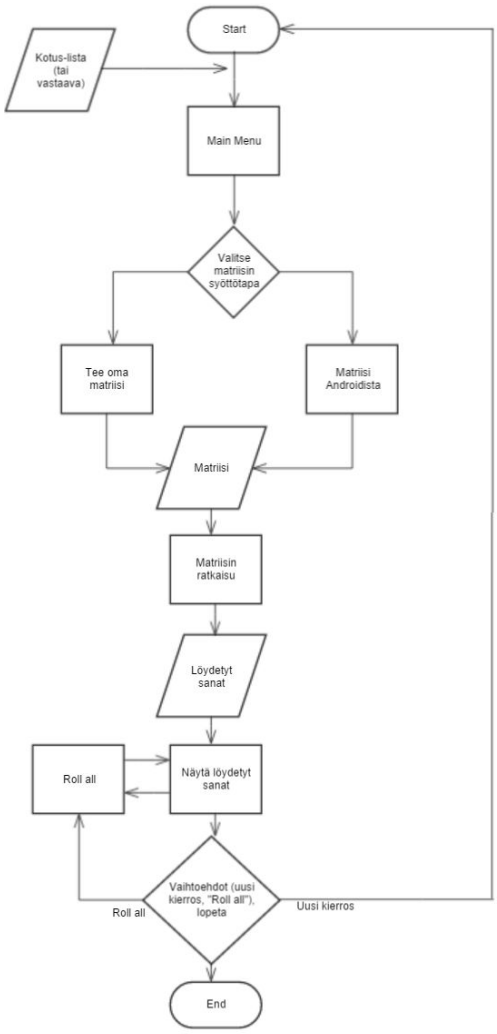
Sanajahti-solver

Lista ominaisuuksista

- **NOPEUS:** Ratkaisee 4x4-kirjainmatriisin alle sekunnissa.
- **JOUSTAVUUS:** Löytyy tuki MxN kokoisille matriiseille
- **TESTATTAVUUS:** Ratkaisualgoritmi on testattavissa. Ratkaisualgoritmia on testattu yksikkötesteillä ja sanajahdin matriiseilla, joista algoritmi löytää kaikki sanakirjasta löytyvät sanat.
- **HYÖDYLLISYYS:** Näyttää myös polun löydettyyn sanaan esim. “rele (0,1) (1,0) (2,0) (1,1)”, jossa (0,0) on vasen yläkulma.
- **MONIKIELISYYS:** Solver toimii useille eri kielille ja merkistöille (kunhan käytettävän kielen sanalista on saatavilla).
- **KÄYTTÖLIITTYMÄ**:
  Tekstipohjainen sekä graafinen; piirtää vastaavan matriisin ja esittää sanat ruudulla SFML:n avulla.
- **ANDROID:** Tuki Androidille tulosten suoraan syöttämiseen. Toteutus Android Debug Bridgellä (ADB). ADB noutaa screenshotin kännykällä ja emuloi näytön kosketuksia.

Ohjelman toiminta: GUI

Käyttäjä avaa ohjelman, jolloin graafisen käyttöliittymän päävalikko käynnistyy (vaihtoehtoisesti ohjelmaa voi käyttää komentoriviltä; lisää alempana). Valikosta käyttäjä voi valita haluaako hän syöttää ratkaistavan kirjainmatriisin itse, vai ratkaistaanko Androidissa käynnissä oleva Sanajahti-peli.



1. **Jos käyttäjä haluaa syöttää ratkaistavan kirjainmatriisin itse**

   Ohjelma kysyy matriisin mitat (MxN), minkä jälkeen se luo käyttäjälle tyhjän ruudukon täytettäväksi. Ruudukon täyttäminen tapahtuu kirjain kerrallaan, alkaen vasemmasta yläkulmasta. Käyttäjän kirjoitettua kirjaimen, kursori siirtyy automaattisesti seuraavaan ruudukon ruutuun. Kun kaikissa ruuduissa on kirjain, käyttäjä voi hyväksyä matriisin ja siirtyä ratkaisuvaiheeseen.

2. **Jos käyttäjä haluaa kirjainmatriisin Androidista**

   Päävalikossa on nappi, jota painamalla ohjelma yrittää tunnistaa matriisin. Jos matriisi tunnistetaan, siirrytään ratkaisuvaiheeseen. Jos matriisia ei löydetä, ohjelma heittää konsoliin exceptionin ja jää päävalikkoon.

Ohjelman ratkaistua syötetyn matriisin, käyttäjälle esitetään kirjainmatriisin vieressä lista löydetyistä sanoista. Käyttäjällä on mahdollisuus klikata yksittäisiä sanoja, jolloin ko. sanan reitti piirretään matriisiin huomiovärillä. Käyttäjällä on myös mahdollisuus vaihtoehtoon “Roll all”, joka esittää yksi kerrallaan jokaisen löydetyn sanan ratkaisun esimerkiksi parin sekunnin välein. Lisäksi, käyttäjä voi valita “Androidiin”, jolloin löydettyjä sanoja aletaan syöttämään mahdollisesti kytkettyyn kännykkään.

GUI:n toiminnasta

GUI luo ikkunan, jonka koko määritellään parametreissä. GUI:n run-metodissa luodaan jokaiselle eri valikolle oma screeninsä (Screen-luokan toteuttavia olioita), jotka tallennetaan mappiin tyyliin <Screen nimi stringinä, screen>. Jokaisella screenillä on run-metodi, joka palauttaa aina seuraavan screenin nimen (tai “exit”:in). Näin GUI kutsuu aina viimeksi palautettua screenin nimeä (alussa käynnistetään “main menu”).

Jokaisen screenin alussa määritellään ko. screenin komponentit osiossa "COMPONENTS INIT". Komponentteja ovat esimerkiksi buttonit ja otsikot. Heti komponenttien määrittelyn jälkeen saatetaan määritellä myös joitain screen-kohtaisia apumuuttujia, tai asettaa sisältöä aiemmin määriteltyihin komponentteihin.

Komponenttien määrittelyn jälkeen tulee varsinainen screenin while-loop, jonka avulla screeniin luodaan responsiivisuus. Tämä while-loop pyörii niin kauan kunnes screen lopetetaan käyttäjän toimesta. While-loopin sisällä käydään läpi tapahtuneet eventit. Tyypillisiä eventejä, joita tarkkaillaan ovat hiiren klikkaus ja liikahdus. Näiden eventien tapahtuessa tarkistetaan tuleeko screeniä vaihtaa, tai jonkin komponentin tilaa muuttaa. Eventien läpikäymisen jälkeen screen renderöidään uudelleen.

GUI määrittelee myös muutaman komponenttityypin. Näitä ovat esimerkiksi Button ja TextField, sekä TextFieldMatrix. Nämä komponentit sisältävät attribuutteja ja metodeja, jotka helpottavat screenien rakennusta. Myös joitakin apumetodeja määritellään GUI:ssä, pääasiassa GUI:n ja solverin yhdistämistä varten.
GUI.hpp:n alussa on määritelty merkistö jota käytetään. Myös fontti määritellään siellä.

Inspiraation lähteitä:
Screenin vaihto:
https://github.com/SFML/SFML/wiki/Tutorial:-Manage-different-Screens
Käytetty fontti:
http://www.1001fonts.com/lazy-opossum-font.html
Buttonin tekstin asettelu:
http://en.sfml-dev.org/forums/index.php?topic=8413.0

Ohjelman toiminta: main()

Pääohjelma aluksi käsittelee sille syötetyt argumentit parse_settings() funktion avulla. Arvot luetaan struct settings -tietorakenteeseen, joka myös asettaa oletusarvot argumenteille. Oletusarvoina OCR, koordinaattien tulostus, ja Android-tuki eivät ole käytössä, matriisin koko on 4x4 ja sanalistana toimii sanat.txt.
Argumenttien lukemisen jälkeen ohjelma lukee sanalistan muistiin read_words_from_file()-funktiolla, ja käynnistää erinäisten if-ehtolauseiden avulla graafisen käyttöliittymän (tai tulostaa konsoliin argumenttien määrittelemillä ehdoilla). Käytettäessä GUI:ta, ohjelman suoritus lakkaa kun GUI suljetaan.

Ohjelman toiminta: shell

Ohjelma tukee kutsuja komentorivistä. Tarkemmat ohjeet löytyvät README.md-tekstitiedostosta.

Ohjelman toiminta: signaalit ja virhetilanteet

Ohjelmassa on pyritty huomioimaan erilaisia virhetilanteita, ja ohjelma tuntee SIGINT-, SIGSEGV- ja SIGTERM-signaalit. Tarkoituksena on, että jos ohjelma kaatuu, syntyisi jokin tulostus tapahtuneesta virheestä (stderr).

Algoritmit

Sanalistojen jaottelu

Sanalistat jaotellaan ohjelman käynnistyessä ensimmäisen kirjaimen mukaan ratkaisunopeuden lisäämiseksi. Tietyllä kirjaimella alkavat sanat tallennetaan eri muuttujiin ja niitä käytetään sanalistan pohjana ratkaisualgoritmissa. Jaottelu toteutetaan ratkaisualgoritmin sisällä.

Matriisin ratkaisu

Matriisin ratkaisu toteutetaan rekursiivisena algoritmina. Kutakin kirjainta matriisissa käytetään vuorollansa alkupisteenä, jonka perusteella valitaan oikea sanalista pohjaksi. Tämän jälkeen etsitään ko. kirjaimen vieressä olevat ruudut. Algoritmissa liikutaan näihin ruutuihin ja sanalistasta etsitään näillä kahdella kirjaimella alkavat sanat. Näissä pisteissä tarkistetaan löytyykö sanaa sanakirjasta, etsitään uudestaan vieressä olevat kirjaimet joissa ei ole käyty, ja rajataan sanakirjasta näillä kirjaimilla alkavat sanat. Tämän jälkeen liikutaan viereisiin ruutuihin ja tarkistetaan samat asiat uudestaan. Tällä tavalla edetään kunnes pienennetyssä sanalistassa ei ole enää sanoja, jolloin kyseinen polku loppuu.

Matriisi ratkaistaan käyttämällä Solver-oliota, joka ottaa parametreikseen sanakirjan, matriisin sanana, sekä matriisin x- ja y-koot. Sana jaetaan x- ja y-koon mukaisesti matriisiksi esim. aaaabbbbccccdddd muuttuu 4x4-kokoiseksi matriisiksi:

aaaa
bbbb
cccc
dddd

Kuvantunnistus (OCR)

Ohjelma noutaa Android Debug Bridgellä (ADB) puhelimesta kuvakaappauksen, josta luetaan puhelimen resoluutio. Resoluutiosta voidaan laskea suhteelliset koordinaatit sanajahdin eri kirjaimille eli “laatoille”. Moottori konfiguroidaan pääohjelman hakemistosta löytyvällä tess_config tekstitiedostolla. Jokaisesta ruudusta luodaan suorakulmio, joka syötetään “tesseract-moottorille” ratkaistavaksi.

Kirjainten tunnistus perustuu moottorissa käytettävään SINGLE_CHAR PageSegModeen, joka määrittää moottorin etsimään vain yhtä kirjainta sille annetusta datasta. Data syötetään tesseractille tarkasti rajattuna suorakulmiona ja moottori antaa ulos kirjaimen. Lopuksi verrataan saadun merkkijonon pituutta odotettuun (16 merkkiä) pituuteen.

OCR toimii vain 4x4 matriisin kanssa ja vain Sanajahdista otetusta kuvakaappauksesta.

Testit

Ratkaisualgoritmia yksikkötestataan Google Test -kirjastojen avulla. Seuraavia asioita testataan:

- Palauttaako solver samat sanat matriisille, sekä sen käännetylle versiolle
- Toimiiko solver eri kielisellä sanalistalla
- Löytääkö solver pitkät sanat
- Kaikkien sanojen löytäminen on testattu lähinnä vertaamalla löydettyjä sanoja sanajahdin sanoihin

GUI:ta testataan kokeilemalla. Seuraavien ominaisuuksien tulee toimia GUI:ta käytettäessä:

- Painikkeet muuttavat väriään punaiseksi kun hiiri laitetaan niiden päälle, ja takaisin siniseksi kun hiiri vedetään pois.
- Main Menu ja Takaisin -napit toimivat nimiensä mukaisesti. Samaten muut painikkeet.
- Matriisin mitoiksi voi syöttää vain numeraaleja, eikä matriisin kirjaimia pääse syöttämään ennen kuin mitat on syötetty.
- Matriisia ei pysty ratkaisemaan ennen kuin jokainen matriisin ruutu on täytetty.
- Ratkaistua sanaa painettaessa sen polku maalataan vihreällä matriisiin.

- Roll All -näyttää jokaisen ratkaistun sanan polun yksitellen matriisissa, ja lopettaa sen kun käyttäjä painaa jotakin ratkaistuista sanoista.
- Android-painike syöttää ratkaistut sanat tietokoneeseen kytkettyyn Android-puhelimeen. Painike muuttaa tällöin nimekseen “Stop”, jonka painaminen pysäyttää syötön.

OCR testataan Google Test -kirjastojen avulla.

- src/test/testpics kansiosta löytyy answers.txt, jossa on vastausstringit yksi per rivi. scrot_0.png vastaus on siis answers.txt ensimmäinen rivi. Lisäämällä kuvia ja vastauksia tekstitiedostoon voidaan testien lukumäärää laajentaa.
- Tunnistus tehdään ocr.hpp:sta löytyvällä ocr() funktiolla.
- Tärkeimpänä on ääkköset ja niiden vaihteleva sijainti. Jotkin muutokset ocr:n rajaussuorakulmioihin (kts. OCR:n toiminta) rikkoivat kulmassa tai reunassa olevat ääkköset.

Huomioitavaa ohjelman ajamiseen liittyen

Ohjelma vaatii joitakin ulkoisia kirjastoja, tarkista lisätiedot ja tarkemmat ohjeet Readme.md:stä.

Toteutunut aikataulu

Viikko	
45	GUI, ratkaisualgoritmi
46	Bugien korjailua, ääkkösten tukeminen, OCR-moottorin “kilpailutusta”
47	Tekstipohjainen käyttöliittymä + Androidiin syöttö
48	GUI:n + muiden ominaisuuksien yhteensovittamista, hiontaa, OCR
49	Android-rajapinta, OCR ja testit
50	Lopullinen versio kaikkine ominaisuuksineen, dokumentointi, joulun odottelua

Henkilökohtaiset työmäärät (arvio)

Nimi	vko 45	vko 46	vko 47	vko 48	vko 49	vko 50
Janne	10h	6h	10h	12h	8h	18h
Tatu	10h	5h	15h	6h	5h	5h
Tomi	22h	6h	5h	2h	3h	5h
Vili	15h	15h	5h	0h	2h	13h

Ryhmän sisäiseen (lähes jokapäiväiseen) kommunikointiin olemme käyttäneet WhatsAppia. Lisäksi olemme pitäneet kolme tapaamista Maarintalolla. Tapaamisissa olemme käyneet läpi projektin tulevaisuutta, yhteisiä linjauksia sekä kirjoittaneet suunnitelmaa tai dokumentaatiota.

Vastuualueet

Janne	OCR, Android, program args
Tatu	Testien pohja, makefilen konfigurointi
Tomi	GUI
Vili	Ratkaisualgoritmi, Android

Jatkokehitysmahdollisuudet

Ohjelmaa voisi jatkokehittää esimerkiksi lisäämällä seuraavat ominaisuudet:

- Ohjelma osaisi poistaa sanalistasta sanoja, joita ei ole käytössä Sanajahti-pelissä. Tämä onnistuu OCR-kirjaston avulla pelin jälkeisestä kuvakaappauksesta, joka näyttää pelissä olleet, ratkaistavat sanat.
- Ohjelma osaisi sulkea mahdolliset mainokset, joita saattaa ilmestyä Android-puhelimeen ratkaisemisen aikana.
- getopt_long -komennot ohjelmalle.

Project repository

https://git.niksula.hut.fi/elec-a7150/sanajahti2