Tuomas Jorma Juhani Räsänen

Stipe: tekstinsyöttömenetelmä perinteisille peliohjaimille

Tietotekniikan
pro gradu -tutkielma
19. toukokuuta 2011

JYVÄSKYLÄN YLIOPISTO
TIETOTEKNIIKAN LAITOS

Jyväskylä

Tekijä: Tuomas Jorma Juhani Räsänen
Yhteystiedot: tuomas.j.j.rasanen@tjjr.fi
Työn nimi: Stipe: tekstinsyöttömenetelmä perinteisille peliohjaimille
Title in English: Stipe: Text Input Method for Dual-Joystick Game Controllers
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 119
Tiivistelmä: Pelikonsolien ja perinteisten peliohjaimien tekstinsyöttömenetelmät ovat pääasiallisesti virtuaalisia valintanäppäimistöjä. Ne ova hitaita ja epämiellyttäviä. Eleisiin perustuvat menetelmät lupailevat huomattavia parannuksia virtuaalisiin näppäimistöihin nähden. Aiempien tutkimusten tulosten perusteella peliohjaimien tekstinsyöttömenetelmät eivät ehkä kuitenkaan ole vielä riittävän nopeita modernien tekstinsyöttötarpeiden tyydyttämiseen. Tässä tutkielmassa esitellään uusi perinteisille peliohjaimille suunniteltu sauvaohjaimien eleisiin perustuva tekstinsyöttömenetelmä, Stipe. Stipe toteutettiin eleitä näppäimistötapahtumiksi tulkitsevana näppäimistöajurina Linux-järjestelmälle. Menetelmän kirjoitusnopeuspotentiaalia selvitettiin pitkän aikavälin käyttäjätutkimuksessa. Koehenkilöt kirjoittivat Stipellä pisimmillään 44 tuntia. He saavuttivat jopa yli 40 wpm:n kirjoitusnopeuden. Käyttäjätestien tulokset osoittavat, että sauvaohjaimien eleisiin perustuvalla menetelmällä on korkea kirjoitusnopeuspotentiaali, mutta menetelmän sujuvan käytön oppiminen on verrattain hidasta.
English abstract: Currently, text input methods for game consoles are mainly virtual selection keyboards. Unfortunately, they are slow and unpleasant to use. Gesture-based methods show promises of significant improvements, but still, they might not be fast enough to satisfy the requirements of modern text entry tasks. In this study, a new gesture-based text input method for dual-joystick game controllers is introduced. It's called Stipe. Stipe was implemented as a Linux keyboard driver which interpretes joystick gestures as keyboard events. Also a long-term usability study was conducted for evaluating the typing speed potential of Stipe. Subjects used Stipe for as long as 44 hours. They reached over 40 wpm typing speed. The results of the usability study show, that Stipe has quite high typing speed potential, but mastering a fluent Stiping skill is comparatively slow process.
Avainsanat: tekstinsyöttö, peliohjain, sauvaohjain, joystick, ele, eletunnistus
Keywords: text input, text entry, game controller, gamepad, joystick, thumbstick, analog stick, gesture, unistroke, recognition

Sisältö

Lyhenteet

WPM Words Per Minute, sanoja minuutissa. Kirjoitusnopeusyksikkö. Sanan pituus on yleisesti määritelty viiden pituiseksi merkkijonoksi sisältäen myös välilyönnit.

CPM Characters Per Minute, merkkejä minuutissa. Kirjoitusnopeusyksikkö.

KSPC Key Strokes Per Character. Kuvaa tekstinsyöttötehtävässä yhden merkin syöttämiseen tarvittujen alkeisoperaatioiden lukumäärää. Nimi tulee näppäimistöjen näppäinpainalluksista, mutta voidaan käyttää myös muunlaisissa menetelmissä.

CKSPC Characteristic Key Strokes Per Character. Tekstinsyöttömenetelmän luonteenomainen KSPC-arvo. Kuvaa kuinka monta alkeisoperaatiota menetelmällä yleisesti ottaen tarvitsee tehdä yhden merkin tuottamiseksi.

MSD Minimum String Distance. Suure, joka kuvaa miten "kaukana" kaksi merkkijonoa ovat toistaan. Toisin sanoen, kuvaa kahden merkkijonon eroavaisuutta. Yksikkönä merkkijonon käsittelyyn tarvittavien alkeisoperaatioiden (lisäys, poisto korvaus) lukumäärä, jotta merkkijonoista saataisiin samat.

OCR Optical Character Recognition. Yleinen termi menetelmille, joilla painettua tekstiä tunnistetaan ja digitalisoidaan.

GPC Gestures Per Character. Vastaava tunnusluku kuin KSPC, mutta GPC lyhennttä käytetään toisinaan eleisiin perustuvissa menetelmissä KSPC:n sijaan.

TotalER Total Error Rate. Kuvaa tekstinsyöttöprosessin kirjoitusvirheiden kokonaismäärää. TotalER ottaa huomioon sekä kirjoituksen aikana tehdyt virheet, että kirjoitettuun tekstiin jääneet virheet.

FOA Foucs Of Attention. Tarkoittaa psykologiassa ja kognitiotieteissä kohdetta, joihin ihminen suuntaa tarkkaavaisuutensa. Esimerkiksi vilkkuva kuvake näyttöruudulla.

HID Human Interface Device. Tietokoneeseen kytkettävä laite, joka nimensä mukaisesti on tarkoitettu ihmisen ja tietokoneen väliseen vuorovaikutukseen.

Kuvat

Taulukot

1 Johdanto

Tekstinsyötöllä (engl. *text entry, text input*) tarkoitetaan niitä toimia, joilla käyttäjä siirtää kielellistä informaatiota tekstuaaliseen muotoon tietokoneen prosessoitavaksi ja taltioitavaksi. Tietokone tulee ymmärtää tässä yhteydessä hyvin laajana käsitteenä. Se kattaa kaikki erilaiset laskentalogiikkaa omaavat laitteet, joiden käyttöliittymissä keskeisessä roolissa on kirjainten, numeroiden tai muiden symbolien tuottaminen. Tekstinsyöttö on siis yksi tärkeimmistä tietokoneiden käyttöliittymien ominaisuuksista.

Tekstinsyöttöominaisuuksia tarvitsevien laitteiden ja niiden toimintaympäristöjen monimuotoisuus on johtanut mitä erilaisimpien tekstinsyöttömenetelmien syntyyn. Työpöytäkoneiden, kannettavien tietokoneiden, älypuhelinten, pelikonsolien ja viime aikoina suosituksi nousseiden kosketusnäytöllisten tablettien tekstinsyöttövaatimukset ovat hyvin erilaiset. Näppäimistöt, puheentunnistus, käsialantunnistus ja erilaiset eleisiin perustuvat menetelmät edustavat vain osaa näiden laitteiden tekstinsyöttömenetelmistä.

Näppäimistö on kannettavien ja työpöytätietokoneiden luultavasti yleisin ja nopein tekstinsyöttömenetelmä. Näppäimistöt ovat intuitiviisia aloittelijoille ja äärimmäisen tehokkaita kokeneiden käyttäjien käsissä. Näppäimistöt vaihtelevat näppäinasetteluiltaan ja fyysisiltä mitoiltaan suuresti. Niitä löytyy täysikokoisista pöytämalleista matkapuhelinten pieniin numeronäppäimistöihin.

Sensoriteknologian läpimurrossa elävät modernit mobiililaitteet käyttävät yhä enenevässä määrin fyysisten näppäimistöjen tilalla erilaisia kosketusnäyttöihin perustuvia tekstinsyöttömenetelmiä. Menetelmät vaihtelevat erikoisemmista elejärjestelmistä perinteisempiin virtuaalisiin QWERTY-näppäimistöihin. Alati Internet-yhteydessä olevat mobiililaitteet ovat tuoneet sosiaalisen median, sähköpostin ja erilaiset verkkopalvelut arjen jokaiseen hetkeen. Uusien palveluiden tuomat haasteet ja teknologiset läpimurrot sekä vaativat että mahdollistavat uusien tekstinsyöttömenetelmien kehittämisen. Mobiililaitemarkkinoilla uusien tekstinsyöttömenetelmien tarve onkin havaittu ja kilpailu uusien menetelmien löytämiseksi on kiivasta.

Pelikonsolit ovat kuitenkin jääneet jälkeen uusia tekstinsyöttömenetelmiä tuottavasta kehitysaallosta. Nykyiset pelikonsolien tekstinsyöttömenetelmät ovat pääsääntöisesti erilaiset näytöllä navigoitavat virtuaaliset näppäimistöt. Ne ovat hitaita ja kömpelöitä. Virtuaaliset näppäimistöt eivät vaadi vain tarkkaa silmä-käsi -koordinaatiota kirjoitusvirheiden minimoimiseksi, vaan ovat myös auttamatta liian hitaita tyydyt-

tämään minkään vaativan tekstinsyöttötarpeen. Pelikonsolivalmistajat ovat pyrkineet ratkaisemaan ongelman muun muassa tarjoamalla erillisinä lisävarusteina näppäimistöjä aina peliohjaimiin liitettävistä pienistä QWERTY-näppäimistöistä täysikokoisiin taustavaloilla varustettuihin QWERTY-näppäimistöihin.

Pelikonsoleissa tarjolla olevan sisällön perusteella yritykset voidaan kuitenkin katsoa epäonnistuneeksi. Siinä missä PC-pelien pikaviestiominaisuudet ovat arkipäivää, pelikonsoleissa niitä ei ole tai niiden käyttö on lähes mahdotonta soveltumattomien tekstinsyöttömenetelmien vuoksi. Pelikonsoleille suunnatut pelit ja hyötyohjelmat ovatkin minimoineet tekstinsyöttötarpeet. Usein näiden ohjelmistojen ainoat tarpeet tekstinsyötölle ovat nimimerkkien tai päivämäärien asettaminen.

Kotien olohuoneissa on myös mahdollista nähdä tulevaisuudessa viihdelaitteiden konvergenssi. Nykyisin olohuoneita sisustavat useat erilliset multimedialaitteet, pelikonsoleista äänentoistolaitteisiin, digivirittimiin ja Internet-yhteydellä varustettuihin televisioihin. Pöytiä ja sohvia koristaa viihdelaitteiden ohjaamiseen tarkoitettu runsaslukuisten kaukosäätimien joukko. Kameroiden, radioiden, musiikkisoittimien ja matkapuhelimien konvergenssin jälkimainingeissa on helppo pitää todennäköisenä vastaavanlaista multimedialaitteiden yhteensulautumista myös kotien olohuoneissa.

Yhteensulautuminen vaatii kuitenkin ohjainteknologian kehittämisen pisteeseen, jossa yhdellä ohjaimella kyetään ohjaamaan langattomasti niin äänenvoimakkuuksia, televisiokanavia kuin pelihahmoa pelimaailmassa. Ohjaimen on mahdollistettava nopea tekstinsyöttö sähköpostin, verkkosurffailun ja sosiaalisen median palveluiden käyttöä varten. Nykyisin olohuoneista löytyvistä ohjaimista peliohjaimet tarjoavat kaikkein houkuttelevimman lähtökohdan kaiken kattavaksi multimediaohjaimeksi. Niistä löytyy tuki langattomalle tiedonsiirrolle ja ne sisältävät useita painikkeita, muita ohjainkomponentteja sekä sensoreita. Ne ovat lisäksi suunniteltu ergonomisilta ominaisuuksiltaan pitkäaikaista käyttöä varten ja soveltuvat siten esimerkiksi verkkosurffaukseen mainiosti. Peliohjaimista puuttuu nykyisellään kuitenkin nopea tekstinsyöttömenetelmä.

Tässä tutkielmassa esiteltävän tutkimuksen tavoitteena oli perehtyä olemassaoleviin tekstinsyöttömenetelmiin ja tekstinsyöttömenelmiä käsittelevään teoriaan sekä kehittää perinteisille kahden analogisen sauvaohjaimen peliohjaimille tekstinsyöttömenetelmä, jonka potentiaalinen kirjoitusnopeus olisi selvästi olemassa olevia tekstinsyöttömenetelmiä suurempi. Tutkimuksen tavoitteena oli toteuttaa ohjelmistoprototyyppi tekstinsyöttömenetelmästä ja suorittaa käyttäjätestejä, joiden avulla saataisiin käsitys kehitetyn menetelmän kirjoitusnopeuspotentiaalista.

1.1 Tutkielman rakenne

Tutkielman teoriaosuus lähtee liikkeelle luvusta 2 perehtymällä kevyesti tekstinsyöttöön liittyvien kognitiivisiin prosesseihin. Luku pyrkii avamaan hieman teoriaa ihmisen kielellisistä prosesseista sekä niihin liittyvien motoristen toimintojen tuottamisesta ja oppimisesta.

Luvussa 3 esitellään tekstinsyöttömenetelmiin liittyviä käsitteitä, luokitellaan tekstinsyöttömenetelmiä niiden taustalla vaikuttavien tekniikoiden perusteella ja luodaan tarvittava teoriapohja tutkielmassa myöhemmin käsiteltävälle tekniselle sisällölle. Luvussa esitellään kirjallisuudessa esitettyjä tunnuslukuja ja metriikoita tekstinsyöttömenetelmien arvioimisen tueksi. Luvun erityisenä painopisteenä on eleisiin perustuvat tekstinsyöttömenetelmät.

Luku 4 käsittelee teoriaa eleiden havaitsemisesta ja tunnistamisesta. Luvussa esitellään eleiden havaitsemiseen liittyviä käsitteitä ja eleiden tunnistamiseen kehitettyjä algoritmeja.

Luvussa 5 tarkastellaan hieman peliohjainten historiaa ja perinteisten peliohjainten tekniikkaa. Luku esittelee yksityiskohtaisesti myöhemmin esiteltävän tekstinsyöttömenetelmän kehitystyössä käytettyjen peliohjainten toimintaperiaatteet ja ohjelmointirajapinnat.

Luku 6 esittelee aiempia tutkimuksia tekstinsyöttömenetelmien kehittämisestä peliohjaimille. Siinä missä luvussa 3 esitellään tekstinsyöttömenetelmiä yleisesti, tässä luvussa esitellään vain peliohjaimille kehitettyjä menetelmiä.

Luvussa 7 esitellään tutkimuksen yhteydessä peliohjaimille kehitetty uusi tekstinsyöttömenetelmä nimeltä Stipe. Luku havainnollistaa yksityiskohtaisesti tekstinsyöttömenetelmän toiminnan konseptitasolla ja esittelee lisäksi Stipen prototyyppitoteutuksen arkkitehtuurin Linux-järjestelmässä. Arkkitehtuurikuvauksessa käydään läpi toteutuksen rakenne komponenteittain.

Luku 8 esittelee Stipen kehityksen yhteydessä tehdyn käyttäjätutkimuksen koeasetelman ja havainnot.

Luvussa 9 poimitaan merkittävimmät havainnot, pohditaan löydöksien merkitystä ja niistä johdettavia suuntia jatkotutkimukselle.

Luku 10 päättää tutkielman yhteenvedolla tutkielmassa esitetyistä löydöksistä ja jatkokehitysideoista.

2 Tekstinsyötön kognitiiviset prosessit

Tekstinsyötöllä (engl. *text input, text entry*) tarkoitetaan tässä tutkielmassa toimia, joilla käyttäjä tuottaa tekstiä tietokoneen käyttöön. Erilaisia tekstinsyöttömenetelmiä on useita. Näppäimistö lienee ilmeisin ja käytetyin tekstinsyöttömenetelmä. Tietokone voidaan ohjelmoida tunnistamaan myös puhetta ja muuttamaan se virkkeiksi, lauseiksi, sanoiksi ja symboleiksi. Myös käyttäjän käsialaa voidaan tulkita tekstiksi tietokoneen toimesta. Yleisesti ottaen tekstinsyöttömenetelmä kuvaa ne käyttäjän ja tietokoneen väliset tapahtumat, jotka mahdollistavat kielellisen informaation siirron käyttäjältä tietokoneelle. Olipa menetelmä mikä tahansa, keskeistä tekstinsyötössä käyttäjän kannalta ovat ne kognitiiviset prosessit, jotka muodostavat käyttäjän sisällä vellovista ajatuksista puhetta ja käsien tai muiden ruuminosien liikkeitä, kieltä. Tässä luvussa esitetään lyhyt teoriakatsaus tekstinsyöttöön liittyvien kognitiivisten prosessien toimintaan ja tarkoitukseen.

2.1 Kieli ja sen tuottaminen

Kieli on kahden osapuolen välisen kommunikaation perusta. Kieli koostuu rajattomasta määrästä lauseita. Lauseet muodostuvat edelleen sanoista, joiden voidaan katsoa olevan pienin merkitystä kantava yksikkö kaikissa luonnollisissa kielissä [44]. Lauseiden sanarakenteen määrää rajallinen määrä sääntöjä. Nämä säännöt muodostavat kielen kieliopin (engl. *grammar*). Kieliopin kaksi tärkeintä tehtävää on erottaa oikeelliset lauseet virheellisistä sekä määrittää erilaisten lauseiden keskinäiset erot [54]. Chomskyn mukaan kielioppi voidaan jakaa kolmeen komponenttiin [37]. *Syntaktinen komponentti* määrittää lauseelle sen kieliopillisen rakenteen. *Fonologinen komponentti* määrittää miten lause tulee ääntää ja *semanttinen komponentti* määrittää lauseen merkityksen. Syntaktinen komponentti siis yhdistää merkityksen ja ääniasun lauserakenteeseen.

Ihminen oppii käyttämään äidinkieltään jo varhaisessa vaiheessa elämäänsä. Alkuun kielelliset toiminnot keskittyvät asioiden ja niitä vastaavien sanojen fonologisten komponenttien yhdistämiseen. Lapsen kuulemat äänneyhdistelmät, sanat, saavat merkityksen. Lapsi oppii hyvin nopeasti myös muodostamaan äänteitä ja ääntämään sanoja. Sanavaraston karttuessa lauseiden ymmärtäminen ja muodostaminen kehittyy. Ihiminen muodostaa siis formaalia kielioppia vastaavan säännöstön mieleensä [54]. On siis ilmeistä, että ihmisellä on jo sisäänrakennettuna jonkinlainen kyky kielelliseen pro-

sessointiin sekä tuottamiseen.

Kielentuottaminen (engl. *language production*) on monivaiheinen ja monia kognitiivisa mekanismeja yhdistelevä prosessi [44]. Se kuvaa ne toimenpiteet, joilla hatarat ja mielessä kelluvat abstraktit ajatukset, tarkoitukset, tunteet ja uskomukset muutetaan ulkomaailmalle tarkoitetuksi kielellisiksi ärsykkeiksi. Ihmisille tavanomaisia kielellisiä ärsykkeitä eli kielen ilmenemismuotoja ovat esimerkiksi puhe, kirjoitus ja viittomat. Ne ovat kaikki perusluonteeltaan tahdonalaista motorista toimintaa vaikka eroavatkin toisistaan suuresti jo fysiologisilta perusteiltaan.

2.2 Tahdonalainen motorinen järjestelmä

Tahdonalaisen motorisen järjestelmän (engl. *voluntary motor system*) ytimenä voidaan kuvitella olevan silmukkarakenne, joka hakee muistista, käynnistää ja ohjaa motorisia komentojonoja vastaanottamiensa ärsykkeiden perusteella [51]. Ärsykkeet luokitellaan sisäisiin ja ulkoisiin ärsykkeisiin. Sisäiset ärsykkeet ovat aikomuksia, tarkoituksia ja haluja muuttaa ulkoisen ympäristön tilaa. Ne syntyvät nimensä mukaisesti henkilön tajunnan sisältä. Ulkoiset ärsykkeet ovat vastaavasti henkilön aistihavaintoja hänen ulkoisesta ympäristöstään. Liike voi olla siis seurausta aistihavainnosta tai erillisestä tarkoituksesta aiheuttaa muutos ympäristössä. Ärsykkeet johtavat ympäristössä muutoksia aiheuttaviin liikkeisiin, jotka edelleen aiheuttavat ärsykkeitä.

Ärsykkeen havaitseminen johtaa informaation tallentumiseen lyhytkestoiseen muistiin, jossa sitä prosessoidaan erinäisten kognitiivisten prosessien voimin. Aistiärsykkeet tuottavat informaatiota jatkuvasti ja siten alati kerääntyvän tiedon määrä on valtava. Ärsykkeiden havaitsemiseen ja prosessointiin liittyy keskeisellä tavalla tarkkaavaisuus (engl. *attention*) ja sen hallinta. Tarkkaavaisuus voidaan käsittää rajallisena kognitiivisena resurssina, jota tarvitaan tehtävien suorittamiseen. Erilaiset tehtävät vaativat tarkkaavaisuutta vaihtelevia määriä.

Tarkkaavaisuutta tarvitaan esimerkiksi aistiärsykkeiden käsittelyyn ja motoristen toimintojen suorittamiseen. Tarkkaavaisuutta vaativia kohteita (engl. *focus of attention*, FOA) voi olla runsaasti. Tarkkaavaisuuden kohdistamista tavoitteen kannalta tärkeisiin tapahtumiin on olemassa hallintaprosessi (engl. *attention control*) [36]. Se voidaan mieltää eräänlaisena suodattimena. Hallitsemalla tarkkaavaisuutta ja sen kohdistamista, ärsyketulvasta voidaan karsia pois kaikki tavoitteen kannalta merkityksetön informaatio. Tarkkaavaisuuden hallinta mahdollistaa sisäisten ja ulkoisten ärsykkeiden tuottaman informaation suodattamisen joko tahdonalaisesti tai refleksinomaisesti. Ilman informaatiotulvan suodattamista työmuisti täyttyisi nopeasti sekalaisella tiedolla häiriten merkittävästi kognitiivisia prosesseja.

2.3 Motoristen taitojen oppiminen

Kuten kaikki oppiminen, myös uuden motorisen taidon oppiminen vaatii muistamista. Muisti voi olla lyhytkestoista, pitkäkestoista tai kaikkea siltä väliltä. Muistiin tallentuminen on siis vaiheittainen prosessi. Mitä useampia toistoja liikesarjasta tekee sitä pidempään se pysyy muistissa [24]. Yhden toiston jälkeen liikesarja säilyy muistissa vain hetken, mutta suuri määrä toistoja mahdollistaa jopa elinikäisen muistamisen.

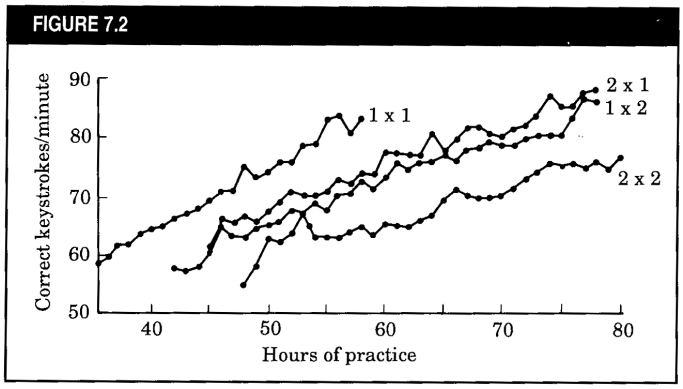
Motoristen taitojen oppiminen voidaan jakaa kolmeen vaiheeseen [53]. Oppimisen ensimmäisessä vaiheessa liikkeille on ominaista suoritusnopeuksien vaihtelu, liikeratojen epäsäännöllisyys sekä niiden kontrollointi pääsääntöisesti aistiärsykkeiden ohjaamana. Keskeistä oppimisessa tässä vaiheessa on löytää oikeat liikkeet yrityksen ja erehdyksen kautta sekä muodostaa perusta motorisen kartan (engl. *motor map*) vahvistumiselle. Oikean liikkeen löytäminen vaatii liikkeen ja sitä saatavan aistiärsykkeen yhdistämistä ja työmuistiin tallentamista. Esimerkiksi käden liikkeitä opeteltaessa ensimmäisiä kertoja, motorinen kartta on heikko. Käden liikettä ohjataan pääasiallisesti asentoärsykkeen ja näköärsykkeen perusteella ja oikean liikkeen suorittaminen vaatii tarkkaavaisuuden keskittämistä liikkeen seuraamiseen sekä käden asentoon.

Oppimisen toisessa vaiheessa liikemuistia kuvaava motorinen kartta vahvistuu asteittain ja sen myötä liikkeiden suoritusnopeudet kasvavat [53]. Aistipalautteen prosessointi työmuistissa ja tarkkaavaisuuden säilyttäminen sekä liikkeessä että sen tarkoituksessa ovat keskeisessä roolissa motorisen kartan vahvistamisessa. Toistamalla liikkeitä motorinen kartta vahvistuu ja siirtyy pitkäkestoiseen muistiin. Mitä vahvempi motorinen kartta on sitä vähemmän prosessointityötä aistiärsykkeet vaativat ja sitä nopeammin motoriset komennot voidaan valita ja suorittaa.

Oppimisen kolmatta vaihetta kuvastavat nopeat ja automaattiset liikkeet [53]. Motoristen komentojen valinta aikomuksen perusteella on nopeaa ja liikkeiden suorittaminen ei vaadi henkilöltä tarkkaavaisuuden osoittamista ulkoisten ärsykkeiden prosessointiin. Motoriseen karttaan on tallentunut liikevasteet sisäisille ja ulkoisille ärsykkeille.

2.3.1 Harjoittelun jaksotus

Käytännön oppimisen kannalta erittäin kiinnostava kysymys on optimaalisten harjoittelujaksojen pituudet ja niiden toistuminen. Tulisiko opettelu tehdä tiiviisti mahdollisimman lyhyessä ajassa vai onko optimaalisempaa jakaa koko opetteluaika useampiin jaksoihin? Baddeleyn ja Longmanin mukaan, yleisesti hyväksytty käsitys on, että harjoittelujaksojen jakaminen pienempiin osiin edistää oppimista [17]. Baddeley ja Longman pyrkivät vahvistamaan tämän käsityksen tutkiessaan konekirjoituksen oppimista postinlajittelijoiden parissa. Kaikki koehenkilöt opettelivat konekirjoitusta yhteensä 80



Kuva 2.1: Konekirjoituksen oppimiskäyrät [17][s. 110].

tuntia. Koehenkilöt muodostivat neljä ryhmää, joiden kokonaisharjoitteluaika jakaantui seuraavasti:

1. 1 tunnin harjoitusistunto kerran päivässä
2. 1 tunnin harjoitusistunto kahdesti päivässä
3. 2 tunnin harjoitusistunto kerran päivässä
4. 2 tunnin harjoitusistunto kahdesti päivässä

Ryhmien oppimiskäyrät on esitettynä kuvassa 2.1. Oppiminen oli nopeinta ryhmässä 1 ja hitainta vastaavasti ryhmässä 4. Tulokset osoittavat, että harjoittelun jakaminen pieniin harjoittelujaksoihin tehostaa oppimista. Baddeley kuitenkin huomauttaa, että vaikka oppiminen suhteessa käytettyyn harjoitteluaikaan on nopeinta jakamalla harjoittelu lyhyisiin harjoittelujaksoihin usean päivän ajalle, niin harjoitusten suorittamiseen käytetty kokonaisaika on tällöin pitkä. Tutkimuksen koehenkilöistä tyytymättömimpiä omaan oppimiseen olivatkin ryhmän 1 koehenkilöt, koska he kokivat edistyvänsä hitaasti harjoitteluun käytetyn kokonaisajan suhteen [16].

2.3.2 Lepo ja harjoittelu

Motoristen taitojen oppimista voi tehostaa harjoittelun jaksottamisen lisäksi levolla. Motoristen taitojen kehittyminen jatkuu vuorokauden ajan harjoittelujakson jälkeen [26]. Walkerin mukaan erityisen merkittävä levon muoto harjoittelun tehostamiseksi on uni [39]. Tutkimuksessa testattiin koehenkilöiden kykyä oppia numeroitujen näppäimien painamista vastaavassa järjestyksessä viiden numeron sarjoissa mahdollisimman nopeasti. Tutkimus osoitti, että yön aikana nukkuessa oppiminen oli merkittävästi nopeampaa kuin päivälevon aikana. Koehenkilöiden suoritukset paranivat viidenneksellä

yöunet sisältävän 12 tunnin levon aikana. Merkittävää suoritusten paranemista ei havaittu lainkaan vastaavan pituisen unettoman levon aikana.

2.3.3 Liikesarjat, ajoitus ja rytmi

Liikkeet muodostavat peräkkäin suoritettuina liikesarjoja. Liikesarjojen tuottamisen kannalta merkittävää on sekä yksittäisten liikkeiden järjestys että niiden ajoitus. Kuten edellisissä luvuissa on todettu, keskeinen motorisen taidon kehittymistä kuvaava piirre on liikkeiden suoritusnopeuksien kasvu. Liikkeiden nopeutuessa ne sulautuvat toisiinsa ja muodostavat liikesarjoja, joiden merkittäväksi ominaisuudeksi muodostuu ajoitus [45].

Allekirjoituksessa käden ja sormien liikkeet toistuvat kerrasta toiseen samankaltaisina samankaltaisella ajoituksella, rytmillä [45]. Käsialan rytmi ei ole kuitenkaan aina sama. Se riippuu kirjoitettavasta tekstistä. Allekirjoituksen rytmi on nimenomaisesti allekirjoituksen tuottavalle motoriselle toiminnalle ominainen. Rytmi muodostuu motorisen harjoittelun kautta ja tallentuu osaksi motorista karttaa [45]. Aivoihimme on siis järjestäytyneenä jonkinlainen mekanismi rytmin prosessointiin ja motoristen toimintojen tuottamiseen tietyssä rytmissä. Toinen esimerkki arkisesta rytmisestä motorisesta toiminnasta on tutun salasanan näppäily tietokoneelle. Tutun salasanan kirjoittaminen on luultavasti jokaisen tietokonekäyttäjän yksi nopeimmista yksittäisistä tekstinsyöttötapahtumista.

Rytmin muodostuminen yhdistetään liikkeiden jaotteluun kognitiivisen prosessoinnille sopiviksi palasiksi (engl. *chunk*) [45]. Liikepalasten suorittaminen on nopeaa ja automaattista, mutta palasten välillä on havaittavissa pidempiä viiveitä kuin palasten sisäosien välillä. Sakai päättelee tämän perusteella motoristen liikkeiden rytmin muodostuvan nimenomaisesti liikesarjojen tallentumisesta palasina [45]. Taidokkaita liikesarjoja kuvaa usein niiden rytmisyys ja automaattisuus.

3 Tekstinsyöttömenetelmät

3.1 Tekstinsyöttömenetelmien luokittelu

Isokoski jakaa erilaiset teksinsyöttömenetelmät kahteen kategoriaan [22]. Valintamenetelmät (engl. *selection*) esittävät käyttäjälle selkeästi rajatun määrän kielirakenteita, joita käyttäjä valitsee muodostaen halutun tekstin. Erinomainen esimerkki tällaisesta valintamenetelmästä on näppäimistö. Se tarjoaa käyttäjälleen kielirakenteina yksittäisiä merkkejä painikkeina, joita painamalla käyttäjä muodostaa halutun tekstikokonaisuuden. Näppäimistöjä käsitellään tarkemmin luvuissa 3.3 ja 3.4.

Toinen kategoria on tunnistusmenetelmät (engl. *recognition*). Ne eivät määrittele tarkkoja valintojen joukkoja, vaan ainoastaan määrittelevät valintoja kuvaavat mallit. Käyttäjän on siis mahdollista tuottaa mielivaltaisia syötteitä, joita tunnistukseen perustuva tekstinsyöttömenetelmä pyrkii tunnistamaan parhaan kykynsä mukaisesti käyttäjän tarkoittamaksi kielirakenteeksi. Tunnistusmenetelmistä esimerkkeinä mainittakoon puheentunnistus, käsialantunnistus ja eletunnistus, joita käsitellään tarkemmin luvuissa 3.5, 3.6 ja 3.7.

Valintamenetelmien ja tunnistusmenetelmien eroa voidaan tarkastella pohtimalla käyttäjän ja tekstinsyöttömenetelmän rooleja tuotettavan kielirakenteen valinnassa. Valintamenetelmissä kielellisen rakenteen valinta on käyttäjän vastuulla, eikä valintamenetelmä jätä lähtökohtaisesti ohjelmistolle tulkinnan sijaa. Valinnat ovat usein erittäin eksplisiittisiä kuten näppäimistön painikkeiden painaminen. Käyttäjä harvoin syyttää tietokonetta virheellisen merkin tuottamisesta, mikäli käyttäjä on itse painanut väärää näppäintä. Väärän näppäimen painaminen on käyttäjän virhe.

Tunnistusmenetelmissä, Isokosken sanoin, käyttäjälle luodaan illuusio vapaudesta tuottaa syötteitä, jotka menetelmän tunnistusalgoritmi tulkitsee kielirakenteiksi [22]. Tunnistusmenetelmissä kielirakenteen valinnan suorittaa tunnistusalgoritmi. Esimerkiksi käsialantunnistuksessa käyttäjä voi kirjoittaa kirjaimia eri tavoin vaihdellen kirjoittamiensa kirjamien kokoa tai tyyliä. Tunnistusalgoritmin tehtävä on tunnistaa kirjaimet oikein niiden ulkoasusta riippumatta. I-kirjaimen tunnistaminen virheellisesti l-kirjaimeksi on lähtökohtaisesti virhe algoritmin toiminnassa, ei käyttäjän käsialassa.

Kolmas tekstinsyöttömenetelmiin kiinteästi liittyvä käsite on kielimallit (engl. *language model*) [22]. Ne jaetaan kahteen kategoriaan [14]. Perinteinen kielimallien kategoria on luvussa 2 esitetyt kieliopit. Ne kuvaavat miten kielessä voidaan yhdistellä

kirjaimista sanoja, sanoista lauseita ja lauseista edelleen suurempia kokonaisuuksia. Toinen tapa mallintaa kieltä on käsitellä sitä tilastollisten menetelmien avulla. Tilastolliset kielimallit voivat kuvaavat todennäköisyyksiä joilla kielirakenteet, esimerkiksi tietyt sanat, esiintyvät peräkkäin [20].

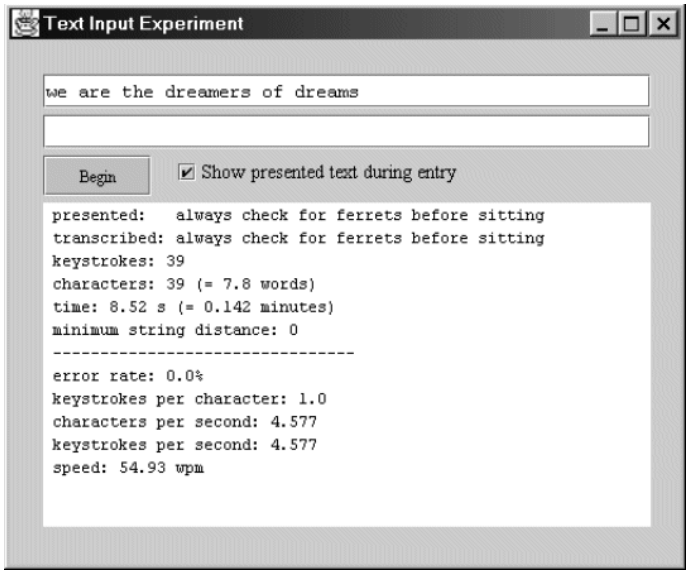
Kielimallit kuvaavat siis kielen rakennetta eri menetelmin. Kielimallien avulla voidaan parantaa tunnistusmenetelmien tunnistustarkkuutta tai helpottaa valintamenetelmissä valintojen tekemistä [22]. Esimerkiksi puheentunnistusmenetelmissä voidaan käyttää sanojen keskinäisten esiintymistodenäköisyyksien suhteita valitsemaan homofonien väliltä todennäköisin vaihtoehto. Näppäimistöissä kielimalleja voidaan käyttää järjestämään näppäimet käyttäjän kannalta suotuisaan järjestykseen [35].

3.2 Tekstinsyöttömenetelmien suorituskyvyn arvioinnista

Tekstinsyöttömenetelmiä tutkittaessa ensimmäinen ja ehkäpä kaikkein tärkein kysymys koskee tutkittavan menetelmän kirjoitusnopeutta. Nopeus on myös ehkäpä helpoin mitattava tekstinsyöttömenetelmän suorituskykyä kuvaavista suureista. On selvää, että tekstinsyötön toinen tärkeä suure on virheiden määrä. Sen mittaaminen on huomattavasti nopeutta monimutkaisempaa. Nopeus ja virheet ovat myös riippuvaisia toisistaan.

Tekstinsyöttömenetelmien suorituskykyä arvoidaan erilaisilla kirjoitustesteillä. Usein kirjoitustestit ovat jäljennöstehtäviä [61]. Jäljennöstehtävissä koehenkilön tehtävä on kopioida annettu teksti tutkittavaa tekstinsyöttömenetelmää käyttäen. Jäljennöstehtävä esitetään koehenkilöille usein kuvan 3.1 kaltaisella yksinkertaisella käyttöliittymällä. Käyttöliittymä sisältää kaksi tekstikenttää, joista alempaan kohdetekstikenttään koehenkilö kirjoittaa ylemmässä lähdetekstikentässä esitetyn tekstin. Testi alkaa usein ensimmäisen merkin ilmestyttyä kohdetekstikenttään tai painamalla erillistä aloituspainiketta. Vastaavasti testi loppuu, kun kohdetekstikenttään on kirjoitettu kopioitava teksti, tai painamalla erillistä lopetuspainiketta. Koehenkilölle voidaan lisäksi näyttää testin lopuksi tilastotietoa testisuorituksesta. Kirjoitusnopeuksien ja virhemäärien esittäminen voi kannustaa ja lisätä motivaatiota testien suorittamiseen [31].

Vaihtoehtoinen koeasetelma jäljennöstehtävälle on vapaan tekstin tuottaminen. Siinä koehenkilölle annetaan vapaus kirjoittaa mitä tahansa tekstiä. Tekstin tuottaminen vapaasti koetilanteessa vaikeuttaa kuitenkin suureiden mittaamista ja tulosten analysointia. Toisaalta, vapaan tekstin tuottamisessa koehenkilön tarkkaavaisuuden kohteita (engl. *focus of attention*, FOA) on yksi vähemmän; koehenkilön ei tarvitse siirtää katsetta kahden tekstikentän välillä tekstinsyötön aikana. Toisaalta, tekstin vapaassa tuottamisessa koehenkilön on keksittävä tekstiä kirjoittaessaan. Kirjoitettavan tekstin keksiminen tekee koetilanteesta vaikeasti hallittavan. Siksi jäljennöstehtävä on suosi-



Kuva 3.1: Tavanomainen käyttöliittymä jäljennöstehtävien esittämiseen [49].

tellumpi vaihtoehto suuremmasta tarkkaavaisuustarpeesta huolimatta [31].

Tekstinsyöttömenetelmien suorituskyvyn arvioinnissa on keskeistä tarkastella tekstinsyöttöä sopivilla tunnusluvuilla. Tekstinsyötön tarkoitus on tuottaa tekstiä nopeasti ja virheettömästi. Nopeuden ja virheiden mittaamiseen käytetään erilaisia tunnuslukuja, joiden käyttöä esitellään seuraavissa aliluvuissa.

3.2.1 Tekstinsyöttönopeus

Ehkäpä kaikkein ilmeisin mittari tekstinsyöttöprosessille ja -menetelmälle on tekstinsyöttönopeus. Tekstinsyöttönopeutta mitataan hyvin yleisesti kirjoitettujen sanojen lukumääränä minuuttia kohden (WPM, *words per minute*) [8, 22, 61]. Sanojen pituusvaihteluiden vuoksi WPM-yksikön sanapituutena käytetään viittä merkkiä, jotka voivat olla kirjaimia, välilyöntejä tai muita välimerkkejä.

Tosiasiallisesti WPM kuvastaa siten kirjoitettujen merkkien lukumäärän *viidesosaa* minuuttia kohden. Toisinaan kirjoitusnopeudet ilmoitetaankin vain kirjoitettujen merkkien lukumääränä minuuttia kohden (CPM, *characters per minute*) [61]. Kirjoitusnopeuden tunnusluvut esitetään täsmällisesti:

$$CPM(T, S) = \frac{|T| - 1}{S} \times 60s \tag{3.1}$$

$$WPM(T, S) = \frac{CPM(T, S)}{5} \tag{3.2}$$

Yhtälöissä käytettyjen suureiden kirjainlyhenteet ovat samoja kuin aihealueen ai-

emmissa julkaisuissa [8, 22, 49, 61]. Yhtälöissä T (*transcribed text*) on tekstinsyötön tuloksena syntynyt merkkijono ja $|T|$ merkkijonon merkkien lukumäärä. Merkkijonon tuottamiseen käytetty kokonaisaika sekunteina on S. Se mitataan ensimmäisen kirjaimen syöttämisestä viimeisen kirjaimen syöttämiseen. Tarkasti ottaen, ensimmäisen kirjaimen syöttämiseen käytettyä aikaa ei mitata ja sen vuoksi merkkijonon pituudesta vähennetään aina yksi merkki [8, 61].

Tekstinsyöttönopeuden mittaaminen WPM:n avulla ei kuvaa tarkasti tekstinsyöttömenetelmää prosessina [61]. Se mittaa ainoastaan lopullisen merkkijonon tuottamiseen käytettyä kokonaisaikaa. Tekstinsyötön aikana voi syntyä virheitä, joiden korjaamiseen kuluu aikaa. Korjatut virheelliset merkit eivät näy lopullisessa merkkijonossa, mutta niiden korjaamiseen kulunut aika on osa yhtälön 3.2 kokonaisaikaa. Näin ollen WPM antaa paremman tuloksen virheelliselle merkkijonolle kuin merkkijonolle, jonka tuottamisen aikana on korjattu virheitä. WPM ei siis huomioi mitenkään annetun merkkijonon P (*presented text*) ja kopioidun merkkijonon välisiä eroja.

Havainnollistetaan tekstinsyöttönopeuden määrittämistä esimerkin avulla:

$$\begin{aligned} P_1 &= \texttt{"kissa istuu puussa"} \quad |P_1| = 18 \\ T_1 &= P_1 \qquad\qquad\qquad\quad |T_1| = |P_1| \quad S_1 = 4,7s \end{aligned} \tag{3.3}$$

$$\begin{aligned} CPM(T_1, S_1) &= \frac{|T_1|}{S_1} \times 60s = \frac{18-1}{4.7s} \times 60s \approx 217,0cpm \\ WPM(T_1, S_1) &= \frac{CPM(T_1, S_1)}{5} \approx 43,4wpm \end{aligned} \tag{3.4}$$

Merkkijonon tekstinsyöttönopeutta voidaan tarkastella tarkemmin pilkkomalla sen tuottamiseen kulunut aika merkeittäin. Merkkijonon tuottamiseen kulunut aika koostuu merkkien välillä vallitsevista viiveistä ja varsinaisista merkkien tuottamiseen kuluvista ajoista. Merkkien tuottamiseen kuluva aika mitataan menetelmästä riippuen hieman eri tavoin. Joissakin menetelmissä rajanveto merkkien välisen viiveen ja merkin tuottamisen välillä on vaikeaa. Esimerkiksi näppäimistöissä merkin tuottamiseen kuluva aika voitaisiin mitata ajaksi näppäimen painamisen ja vapauttamisen välillä. Oikeampi menetelmä olisi kuitenkin mitata lisäksi se aika, jonka aikana sormi liikkuu näppäimen kohdalle [61].

Merkkien välisten viiveiden ja merkkien tuottamiseen kuluvan ajan välillä on kuitenkin merkittävä ero. Varsinainen merkkien tuottamiseen kuluva aika kertoo enemmän tekstinsyöttömenetelmän alkeisoperaatioiden vaikeudesta. Mikäli tekstinsyöttömenetelmällä yksittäisenkin merkin tuottaminen on hidasta, ei menetelmä voi olla kovin nopea tuottamaan tekstiä myöskään suuremmissa määrin. Viiveet taas kuvaavat enemmänkin menetelmän kognitiivisille prosesseille asettamaansa kuormaa. Pitkät viiveet merk-

kien välillä voivat kertoa menetelmän oppimisen vaikeudesta tai vaikeudesta muistaa merkkejä vastaava liike [57].

3.2.2 Kirjoitusvirheet

Kirjoitusvirheiden yksiselitteinen mittaaminen on huomattavasti tekstinsyöttönopeuden mittaamista vaikeampaa. Kirjoitusprosessin aikana syntyneiden ja korjattujen virheiden määrää voidaan mitata laskemalla näppäinpainallusten lukumäärää suhteessa lopullisen merkkijonon pituuteen ($KSPC$, *keystrokes per character*) [8, 30, 49, 61].

$$KSPC(IS, T) = \frac{|IS|}{|T|} \tag{3.5}$$

Yhtälössä IS on syötevirta (engl. *input stream*), joka sisältää kaikki merkkijonon T tuottamiseen johtaneet näppäinpainallukset askelpalautukset (engl. *backspace*) mukaan lukien. Vaikka suureen kirjainlyhenne $KSPC$ viittaa näppäimistöihin, sitä voidaan käyttää yhtä hyvin kaikissa tekstinsyöttömenetelmissä, joissa syötevirta voidaan kuvata yksittäisiä merkkejä tuottavina alkeistapahtumina [30, 61]. Toisinaan eleisiin perustuvissa tekstinsyöttömenetelmissä voidaan käyttää vastaavaa metriikkaa, jossa näppäinpainallusten sijaan mitataan suoritettujen eleiden lukumäärää [59]. Tällöin suureesta käytetään kirjainlyhennettä GPC (*gestures per character*).

Olkoon T_1, kuten on yhtälössä 3.3 esitettynä ja

$$IS_1 = \texttt{"kisk<sa istyu<<uu pus<ussa"} \quad |IS_1| = 26 \tag{3.6}$$

Esimerkissä <-merkit kuvaavat askelpalautuksia. Tekstinsyötölle on nyt laskettavissa näppäinpainallusten lukumäärän ja tuotetun merkkijonon pituuden suhde seuraavasti:

$$KSPC(T_1, IS_1) = \frac{|IS_1|}{|T_1|} = \frac{26}{18} \approx 1,45 \tag{3.7}$$

Kirjoitusvirheistä johtuen esimerkissä on siis tehty keskimäärin 1,45 näppäinpainallusta jokaista kirjoitettua merkkiä kohden.

Tekstinsyöttömenetelmälle voidaan myös määrittää nk. luonteenomainen KSPC-arvo (engl. *characteristic KSPC*, CKSPC). Se kertoo kuinka monta alkeisoperaatiota tekstinsyöttömenetelmä vaatii yhden merkin tuottamiseksi keskimäärin. Esimerkiksi QWERTY-näppäimistöllä yhden kirjaimen tuottaminen vaatii aina vain yhden näppäinpainalluksen, koska jokaista kirjainta vastaa oma näppäimensä. QWERTY-näppäimistön CKSPC-arvo on siis 1,0. Matkapuhelimien perinteisillä 12 näppäimen näppäimistöillä CKSPC-arvo on $>$1,0. Ennustavien tekstinsyöttömenetelmien CKSPC-arvot voivat olla $<$1,0.

Sekä KSPC että CKSPC mittaavat siis tietyssä mielessä kirjoitustehokkuuksia. Luonteenomainen KSPC-arvo kuvaa kuinka työläs tekstinsyöttömenetelmä on yleisesti merkkijonojen tuottamisessa. Yhtä tekstinsyöttötapahtumaa kuvaava KSPC-arvo kuvaa kuinka paljon työtä, menetelmän alkeisoperaatioita, merkkijonon tuottamiseen on tarvittu tehdä. Se ei kuitenkaan kerro mitään tuotetun merkkijonon laadusta. Virheellisenkin merkkijonon tuottaminen ilman korjaustoimenpiteitä tuottaa pienen KSPC-arvon.

Kopioinnin tarkkutta, eli annetun ja tuotetun merkkijonon eroavaisuutta, mitataan etsimällä pienin merkkijonojen välinen etäisyys (MSD, *minimum string distance*) [48]. Pienimmän etäisyyden löytämiseksi käytetään Levenshteinin algoritmia [29]. Wobbrocking esittämä pseudokooditoteutus on listattuna liitteessä B.2 [61].

MSD lasketaan suorittamalla joukko muunnosoperaatioita lähdemerkkijonon muuntamiseksi kohdemerkkijonoksi. Merkkijonojen välinen etäisyys $MSD(P,T)$ on muunnokseen tarvittavien operaatioiden lukumäärä. Merkkijonolle tehtävä muunnosoperaatio voi olla yksittäisen merkin **lisäys**, **poisto** tai **korvaus** jollain toisella merkillä.

$$\begin{aligned} P_1 &= \texttt{"kissa istuu puussa"} \quad |P_1| = 18 \\ T_2 &= \texttt{"kiska isuu puussda"} \quad |T_2| = 18 \\ &= \quad \texttt{\^{}} \quad \texttt{\^{}} \quad\quad \texttt{\^{}} \end{aligned} \tag{3.8}$$

Kopioitu merkkijono T_2 sisältää annettuun merkkijonoon P_1 nähden virheitä, jotka on osoitettu alimmalla rivillä `^`-merkillä. Ensimmäinen virhe voidaan korjata korvaamalla `k`-kirjain `s`-kirjaimella. Toinen virhe korjautuu helpoimmin lisäämällä `t`-kirjain `u`-kirjaimen kohdalle. Kolmannen virheen korjaamiseen käytetään ylimääräisen `d`-kirjaimen poistamista. Merkkijonon T_2 korjaamiseen vaaditaan yhteensä kolme operaatiota. Merkkijonojen P_1 ja T_2 pienin etäisyys on

$$MSD(P_1, T_2) = 3 \tag{3.9}$$

Merkkijonojen välistä pienintä etäisyyttä käytetään edelleen laskettaessa suhteellinen virheiden osuus ($MSDER$, *minimum string distance error rate*) kopioidussa tekstissä [48, 61].

$$MSDER(P,T) = \frac{MSD(P,T)}{MAX(|P|,|T|)} \times 100\% \tag{3.10}$$

Edellisen esimerkin tapauksessa suhteellinen virheiden osuus on:

$$MSDER(P_1,T_2) = \frac{MSD(P_1,T_2)}{MAX(|P_1|,|T_2|)} \times 100\% = \frac{3}{18} \approx 16,7\% \tag{3.11}$$

Lyhenne	Engl.	Merkitys
C	*Correct*	Oikeat merkit
INF	*Incorrect, Not Fixed*	Korjaamattomat virheelliset merkit
IF	*Incorrect, Fixed*	Korjatut merkit
F	*Fixes*	Korjaustapahtumat

Taulukko 3.1: Syötevirran merkkiluokat.

Toisin kuin $KSPC$, MSD ja $MSDER$ mittaavat vain lopullisen merkkijonon virheellisyyttä. Ne eivät siis kuvaa kirjoitusprosessin tai menetelmän yleistä virheherkkyyttä. Käyttäjä voi tehdä kirjoittaessa valtavan määrän virheitä, mutta mikäli ne korjataan kirjoituksen aikana, MSD ja $MSDER$ osoittavat merkkijonon olevan virheetön [8].

Sekä kirjoituksen aikana tehtyjen että lopputulokseen jäänneiden virheiden määrän mittaamiseksi on kehitetty virheiden kokonaismäärää arvioiva mittari ($TotalER$, *total error rate*) [49]:

$$TotalER = \frac{|INF| + |IF|}{|C| + |INF| + |IF|} \times 100\% \tag{3.12}$$

$TotalER$ luokittelee syötevirran IS jokaisen merkin taulukossa 3.1 esitettyihin neljään luokkaan [49]. Korjaustapahtumat (F) käsittävät kaikki sellaiset syöttötapahtumat, joilla muokataan kirjoitettua merkkijonoa. Korjaustapahtumia ovat esimerkiksi askelpalautukset ja tekstinavigoinnit.

Korjatut merkit (IF) eivät näy lopullisessa merkkijonossa T. Niitä ovat merkit, jotka on poistettu korjaustapahtuman johdosta ja mahdollisesti korvattu uudella merkillä. Korjatut merkit ovat usein kirjoitusvirheitä, mutta ne voivat olla myös annettuun merkkijonoon P nähden oikein merkkejä. Ne siis edeltävät syötevirrassa aina korjaustapahtumia.

Korjaamattomat virheelliset merkit (INF) ovat lopulliseen merkkijonoon T jääneitä kirjoitusvirheitä. Niitä ovat niin väärät merkit, ylimääräiset merkit kuin puuttuvat merkitkin. Korjaamattomien virheellisten merkkien lukumäärä $|INF|$ on aina yhtäsuuri kuin annetun merkkijonon P ja tuotetun merkkijonon T välinen pienin etäisyys:

$$|INF| = MSD(P, T) \tag{3.13}$$

Oikeat merkit (C) ovat merkkejä, jotka ovat annettuun merkkijonoon P nähden oikeita. Oikeiden merkkien lukumäärä $|C|$ on:

$$|C| = MAX(|P|, |T|) - |INF| \tag{3.14}$$

Havainnollistetaan syötevirran IS_2 tapahtumien luokittelua esimerkin avulla. Olkoon P_1 ja IS_2:

$$\begin{aligned} P_1 &= \texttt{kissa istuu puussa} \\ IS_2 &= \texttt{kisk<sa isto puus} \\ T_3 &= \texttt{kissa isto puus} \end{aligned} \tag{3.15}$$

Tällöin syötevirran IS_2 tapahtumat luokitellaan seuraavasti:

C			IF	F	C						INF		C				INF	
k	i	s	k	<	s	a		i	s	t	o		p	u	u	s		

$$\begin{aligned} |C| &= 13 \quad |INF| = 4 \\ |IF| &= 1 \qquad\quad |F| = 1 \end{aligned} \tag{3.16}$$

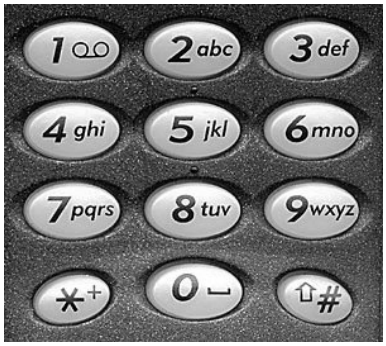
Kirjoituksen kokonaisvirheet määräytyvät siten seuraavasti:

$$TotalER = \frac{|INF| + |IF|}{|C| + |INF| + |IF|} \times 100\% = \frac{4+1}{13+4+1} \times 100\% \approx 27,8\% \tag{3.17}$$

3.3 Fyysiset näppäimistöt

Näppäimistöt lienevät kaikkein yleisin tekstinsyöttömenetelmä. Näppäimistöillä viitataan tässä tutkielmassa nyt ja jatkossa fyysisiin näppäimistölaitteisiin. Ne tulee erottaa ohjelmallisesti näytölle piirretyistä virtuaalisista näppäimistöistä (engl. *virtual keyboards, soft keyboards*), jotka esitellään seuraavassa luvussa 3.4. Näppäimistöjen käyttöä tietokoneiden tekstinsyöttölaitteena voidaan pitää luonnollisena jatkumona mekaanisille ja sähköisille kirjoituskoneille. Kirjoituskoneista tuttu QWERTY-näppäimistöasettelu on niin ikään säilynyt vallitsevana näppäimistöasetteluna myös tietokoneiden näppäimistöissä [46].

QWERTY-näppäimistöjen tekstinsyöttönopeudet vaihtelevat suuresti kohderyhmästä riippuen [22]. Ammattimaiset kirjoittajat kykenevät kirjoittamaan jopa yli 100 wpm:n nopeudella, kun keskimääräisen käyttäjän kirjoitusnopeus on suuruusluokkaa 40 wpm. Maailman nopeimmat kirjoittajat voivat kirjoittaa hetkittäin yli 200 wpm:n nopeudella.



Kuva 3.2: Perinteinen matkapuhelimen 12 näppäimen näppäimistö.

Näppäimistöjä on täysikokoisten työpöytänäppäimistöjen lisäksi hyvin useita erilaisia. Matkapuhelimissa on 1980-luvun ensimmäisistä malleista asti käytetty kuvassa 3.2 esitettyä 12 näppäimen näppäimistöä [46]. Kirjaimet ovat sijoitettuina painikkeisiin 2-9 aakkosjärjestyksessä. Tällaisen näppäimistön tavallisin tekstinsyöttömenetelmä on niin kutsuttu monipainallusmenetelmä (engl. *multitapping*). Kirjain valitaan painamalla numeropainikketta yhden tai useamman kerran. Painallusten lukumäärä vastaa kirjaimen sijaintia painikkeen "sisällä". Esimerkiksi e-kirjaimen tuottaminen vaatii numeropainikkeen 3 painamista kaksi kertaa. Pahimmassa tapauksessa painiketta pitää painaa jopa neljä kertaa yhden kirjaimen tuottamiseksi. Monipainalluksen CKSPC-arvo onkin 2,0342 [30]. Isokosken mukaan monipainallusmenetelmän tyypillinen kirjoitusnopeus aloittelijoilla on 7 wpm:n luokkaa [22]. MacKenzien tutkimuksessa 20:n yliopisto-opiskelijan alkunopeus monipainallusmenetelmällä oli 7,2 wpm. Harjoitusjakson (20×25min) lopuksi joukon keskiarvo oli 15,5 wpm [32].

Monipainalluksen puutteita korjaamaan on kehitetty sanakirjatäsmennykseen (engl. *dictionary-based disambiguation*) perustuvia menetelmiä [30, 46] [1]. Kirjoitettavan tekstin jokaista kirjainta kohden painetaan vain kerran kirjainta vastaavaan numeropainiketta. Tuotettua numerosarjaa verrataan puhelimen muistiin tallennettun sanakirjan sanojen vastaaviin numerosarjoihin. Useat eri sanat voivat tuottaa saman numerosarjan ja sanakirjan tarjoaman sanan voi joutua valitsemaan eksplisiittisesti valikosta useiden vaihtoehtojen joukosta. Sanakirja ei myöskään sisällä välttämättä haluttua sanaa, jolloin sana tulee lisätä sinne ensimmäisen esiintymisen yhteydessä ja kirjoittaminen hidastuu hieman. Mikäli sanakirjan sanasto ja kirjoitettava teksti kohtaavat, menetelmän CKSPC-arvo on 1,0072 [30]. Ihannetilanteessa näppäinpainallusten lukumäärä on selvästi monipainallusmenetelmää pienempi ja 20 wpm:n kirjoitusnopeus on helposti saavutettavissa [46].

Osassa uudemmista älypuhelimista käytetään pienikokoista QWERTY-näppäimistöä.

[1] Tunnetuin sanakirjatäsmennykseen perustuva menetelmä lienee T9 (www.t9.com).

Niiden painikevalikoima sisältää tavallisimmin kirjaimet, numerot, yleisimmät erikoismerkit ja muutamat muut erikoispainikkeet. Suurin ero täysikokoisiin QWERTY-näppäimistöihin ei ole niinkään näppäinvalikoima vaan näppäinten koko ja asettelu. Pienen kokonsa vuoksi näitä niin kutsuttuja mini-QWERTY -näppäimistöjä käytetään usein vain kahdella peukalolla [46]. Kahden peukalon menetelmällä kirjoitusnopeuden teoreettinen maksimi on 60,74 wpm [33]. Isokoski arvioi kokeneiden käyttäjien todellisen kirjoitusnopeuden olevan kuitenkin lähempänä nopeuksia 20-40 wpm [22]. Clarksonin tutkimuksessa lähes kokemattomat käyttäjät kirjoittivat keskimäärin 31 wpm:n nopeudella ja vajaan seitsemän tunnin harjoittelun jälkeen heidän kirjoitusnopeutensa oli keskimäärin yli 60 wpm [15].

Näppäimistöjen voitaneen väittää olevan perusolemukseltaan äärimmäisen intuitiivisia. Ne koostuvat vaihtelevasta määrästä yksinkertaisia painikkeita. Painikkeiden päälle on painettu kuvat niistä symboleista, jotka ilmaantuvat tietokoneen näytölle näppäintä painettaessa. Näppäimistöjä voidaan siten pitää selkeästi valintamenetelmänä, joilla kokematonkin kirjoittaja kykenee tuottamaan tekstiä lähes ilman opettelua. Näppäimistöt tarjoavat oppimiselle helpon polun, jonka päässä häämöttää tekstinsyöttönopeus vailla vertaa. Kokeneet kymmensormimenetelmän taitajat kykenevät kirjoittamaan siirtämättä katsetta pois syntyvästä tekstistä. Heiltä kirjoittaminen onnistuu jopa silmät kiinni. Näppäimistöt ovatkin niin sanottuja katsevapaita (engl. *eyes-free*) menetelmiä. Näppäimistöjen itsestään selvä varjopuoli on kuitenkin se, että ne vaativat aina tietyn fyysisen tilan, jota ei kaikissa laitteissa ja käyttökohteissa ole mahdollista tarjota.

3.4 Virtuaaliset näppäimistöt

Virtuaaliset näppäimistöt ovat näyttölaitteille piirrettäviä näppäimistöjä, joissa näppäinvalinnat tehdään osoitinlaitteiden avulla. Esimerkkejä ovat kosketusnäytöille piirrettävät näppäimistöt, joiden näppäimiä painellaan sormilla tai osoitinkynällä. Pelikonsolien virtuaalinäppäimistöillä näppäinpainallukset tehdään navigoimalla osoitinta peliohjaimien avulla.

Kosketusnäytöllisten laitteiden, kuten modernien älypuhelinten ja kämmentietokoneiden, tavanomainen tekstinsyöttömenetelmä on virtuaalinen näppäimistö. Virtuaalisten näppäimistöjen suosion syynä on niiden intuitiivisuus. Virtuaaliset näppäimistöt ovat vain näytölle piirrettäviä kuvia fyysisistä sisaruksistaan ja ne eivät siksi vaadi alkuun juurikaan uutta opeteltavaa. Käyttötaidot siirtyvät fyysisistä näppäimistöistä virtuaalisiin näppäimistöihin [22].

Virtuaalisissa näppäimistöissä osoitinlaitteen liikettä kohti valittavaa näppäintä oh-

jataan näköaistin avulla. Virtuaaliset näppäimistöt eivät tarjoa sormille tuntoaistimuksia, kuten fyysiset näppäimistöt tekevät. Virtuaalisilla näppäimistöillä kirjoittaminen vaatii siten tarkkaavaisuuden kohdistamista vuoroin näppäimistöön ja vuoroin tuotettavaan tekstiin. Erityisesti suhteellisilla ohjainlaitteilla, kuten pelikonsolien sauvaohjaimilla, näppäimistön spatiaalinen hahmottaminen ilman näköaistia on mahdotonta.

Toisin kuin fyysisissä näppäimistöissä, virtuaalisten näppäimistöjen näppäinten sijainteja ei voi tunnustella sormilla. Virtuaalisten näppäimistöjen näppäinvalinnat perustuvat pääasiallisesti näköaistiin. Erityisesti pelikonsolien virtuaaliset näppäimistöt vaativat jatkuvaa silmä-käsi -koordinaatiota. Niissä valinta tehdään suhteellisilla osoitinlaitteilla, esimerkiksi liikuttelemalla osoitinta sauvaohjainta kääntämällä. Sormien hyvää kykyä spatiaaliseen hahmottamiseen on mahdotonta käyttää hyväksi.

3.5 Puheentunnistus

Puheentunnistus tekstinsyöttömenetelmänä on kiehtova tutkimuskohde. Puhe on ihmiselle kaikkein luonnollisin kommunikaatiomenetelmä ja siten sen käyttö myös tekstinsyöttömenetelmänä on luonnollinen kiinnostuksen kohde. Puhuminen on myös nopea tapa siirtää informaatiota. Suurin osa ihmisistä kykenee puhumaan helposti yli 100 wpm:n nopeudella, mutta virheettömästi tunnistettavan tekstin sanelunopeus on vain 14 wpm [25, 46].

Puheentunnistus voi toimia tarkkuusongelmista huolimatta hyvänä vaihtoehtomenetelmänä, kun perinteiset tekstinsyöttömentelmät eivät ole riittäviä. Esimerkiksi liikuntarajoitteiset potilaat voivat tuottaa puheentunnistuksen avulla tekstiä näppäimistöjen tai käsinkirjoituksen sijaan [52]. Kuten ei mikään muukaan yksittäinen menetelmä sovellu jokaiseen tilanteeseen, myös jopa ihanteellisen tarkalla puhteentunnistusmenetelmällä on varjopuolet. On helppoa kuvitella tilanteita, jolloin ääneen tietokoneelle, mobiililaitteelle tai pelikonsolille puhuminen ei ole mahdollista. Ruuhkaisissa junissa yksityisen sähköpostin sanelu voi olla kiusallista, ainakin kanssamatkustajien kannalta.

3.6 Käsialantunnistus

Käsialantunnistus (engl. *handwriting recognition*) on tekstinsyöttömenetelmä, jossa käyttäjän kirjoittama teksti syötetään ohjelmistolle tunnistettavaksi. Kirjoitus voidaan syöttää prosessoitavaksi esimerkiksi skannaamalla kuva tuotetusta tekstistä tai erillisellä kynällä, jonka liikeradat tunnistusohjelmisto havaitsee liikkeen aikana. Näistä edellinen perustuu ainoastaan jälkikäteen suoritettavaan kirjainten avaruudelliseen (engl.

spatial) tarkasteluun, jota kutsutaan myös optiseksi kirjainten tunnistukseksi (engl. *optical character recognition, OCR*). Jälkimmäisessä menetelmässä käytetään kirjainten geometristen ominaisuuksien tarkastelun lisäksi niiden ajallista (engl. *temporal*) tarkastelua. Kirjainten ajallisia ominaisuuksia ovat esimerkiksi piirtovetojen (engl. *stroke*) lukumäärät, suunnat ja nopeudet [46]. Ajalliseen tarkasteluun perustuva käsialantunnistus on menetelmiltään samankaltaista kuin eletunnistus ja se käsitellään siksi tarkemmin tässä tutkielmassa seuraavassa luvussa elementelmien yhteydessä 3.7.

Käsialantunnistus tekstinsyöttömenetelmänä on luonnollinen mutta hidas. Käsinkirjoitusnopeus on yleisesti ottaen luokkaa 16 wpm [11]. Käsialantunnistus on siis selvästi esimerkiksi näppäimistöjä hitaampi tekstinsyöttömenetelmä.

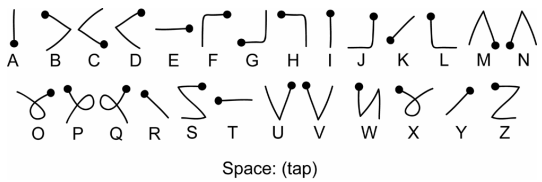
3.7 Eleisiin perustuvat menetelmät

Eleisiin perustuvat menetelmät eroavat ajalliseen tarkasteluun perustuvasta käsialantunnistuksesta aakkostonsa puolesta. Käsialantunnistuksella ymmärretään käyttäjän luonnollisen kielen aakkostoa käyttävän kirjoituksen tunnistamista. Eletunnistusmenetelmissä eleet on määritelty erillisillä eleaakkostoilla, jotka voivat poiketa merkittävästikin luonnollisen kielen aakkostoista. Eleaakkostot ovat yksinkertaisesti vain eleiden ja niitä vastaavien merkkien kuvauksia. Ne siis yhdistävät eleen kuvaukset merkityksiin. Käyttäjälle esitettävä eleaakkosto onkin usein vain joukko eleiden kuvia ja niitä vastaavia kirjaimia tai merkkejä. Esimerkki tällaisesta eleaakkostosta on kuvan 3.3 eleaakkosto.

Eleaakkoston eleet voivat olla ikonisia, jolloin ne muistuttavat luonnollisen kielen aakkosia. Esimerkiksi kuvan 3.3 eleistä L-kirjaimen ele on hyvin ikoninen. Ikonisuuden edesauttaa luonnollisesti eleaakkoston opettelua.

Elemenetelmien eleet voivat olla yksivetoisia (engl. *unistroke*) tai monivetoisia (engl. *multistroke*). Vedolla tarkoitetaan tässä yhtä yhtenäistä liikettä. Esimerkiksi luonnollisen aakoston kirjaimista l-kirjain on yksivetoinen. Se tehdään yhdellä yhtenäisellä liikkeellä. H-kirjain on vastaavasti monivetoinen. Se piirretään kolmesta erillisestä osasta ja osien välillä kynä tulee nostaa ja siirtää uuteen paikkaan. Sen piirtäminen ei muodostu yhtenäisestä kynän vedosta.

Parhaimillaan yksivetoisilla elemenetelmillä voidaan vähentää kirjoittamiseen vaadittavien tarkkaavaisuuden kohteiden lukumäärää. Käyttäjän ei tarvitse seurata katseella kynän liikettä, koska liikkeen suoritus ei ole riippuvainen aiemmin tuotetuista eleistä eikä kynän absoluuttisesta paikasta. Käyttäjä voi tehdä eleen ilman seuraamista ja katse voi keskittyä tuotettavaan tekstiin, aivan kuten kymmensormijärjestelmän osaajat näppäimistöillä.



Kuva 3.3: Unistrokes-eleaakkosto.

3.7.1 Unistrokes

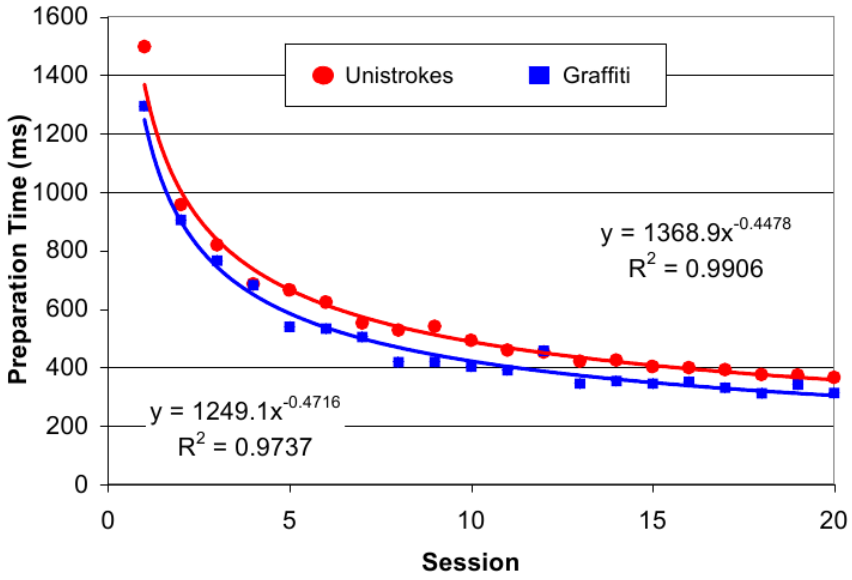
Unistrokes on Goldbergin ja Richardsonin kehittämä ja vuonna 1993 julkaisema eletunnistukseen perustuva tekstinsyöttömenetelmä kynäkäyttöliittymiin [19]. Unistrokesin kantavana ideana oli käyttää eleaakkostoa, joka olisi aloittelijan helppo omaksua ja palkitsisi myös kokeeneen käyttäjän käsialantunnistusta nopeammalla tekstinsyöttönopeudella. Aiemmat kynäkäyttöisten laitteiden tekstinsyöttömenetelmät olivat perustuneet vain käsialantunnistukseen ja eivät Goldbergin ja Richardsonin mukaan tarjonneet kokeneelle käyttäjälle niitä hyötyjä, joita yksinkertaistetulla kuvan 3.3 eleaakkostolla voisi tarjota. Unistrokesin eleet ovat nopeampia kirjoittaa, niiden yksinkertaisuus helpottaa tunnistamista ja yksivetoisuus mahdollistaa katsevapaan kirjoitustavan.

Castellucci ja MacKenzie tutkivat Unistroksin kirjoitusnopeutta (WPM), korjaustiheyttä (engl. *correction rate*, CR), yksittäisten eleiden suoritusnopeuksia, viiveitä eleiden välillä ja aikoja, joita koehenkilöt käyttivät heidän eteensä asetetun eleaakkoston tarkasteluun lauseiden aikana [12]. Virhemääriä mittaavaa korjaustiheyttä mitattiin askelpalautusten lukumääränä ($|F|$) suhteessa kirjoitettavan lauseen pituuteen ($|T|$) ja ilmoitettiin siten prosenttilukuna:

$$CR = \frac{F}{|T|} \times 100\% \tag{3.18}$$

Tutkimus kattoi yhteensä 20 × 15 min testikertaa 10 koehenkilölle. Koehenkilöt eivät olleet kirjoittaneet menetelmällä aiemmin.

Koehenkilöt kirjoittivat ensimmäisellä testikerralla nopeudella 4,1 wpm (σ=2,18) ja viimeisellä 15,8 wpm (σ=4,02) . Ensimmäisen testikerran virhemäärä oli keskiarvoisesti 43,4% ja viimeisen 16,3%. Catellucci ja MacKenzie selittävät korkeita virhemääriä sillä, että käyttäjät eivät usein huomanneet virheitä heti niiden synnyttyä, vaan vasta kirjoitettuaan lausetta muutamia merkkejä eteenpäin. Näin olleen he palasivat virheellisen merkin kohdalle pyyhkimällä myös sen jälkeen kirjoitetut mahdollisesti oikeat merkit. He havaitsivat myös käyttäjien korjaavan usein monta kertaa ongelmallista elettä. Käyttäjät tuntuivat yrittävän muistella ja arvata kirjainta vastaavaa elettä mielummin, kuin tarkistaa sen heidän edessään olevasta aakkostosta. Ensimmäisen testikerran aikana koehenkilöt käyttivät keskimäärin 12,5 (σ=23,6) sekuntia eleen etsimiseen



Kuva 3.4: Unistrokes- ja Graffiti-eleiden viiveet [12].

eleaakkostosta.

Kuvat 3.5 ja 3.4 esittävät Unistrokes-eleiden suoritusajat sekä viiveet. Yksittäisten eleiden suorittamiseen käytetty aika ensimmäisellä testikerralla oli noin 350 ms ja viimeisellä <250 ms. Viiveet eleiden välillä olivat ensimmäisellä testikerralla 1500 ms ja viimeisellä <400 ms.

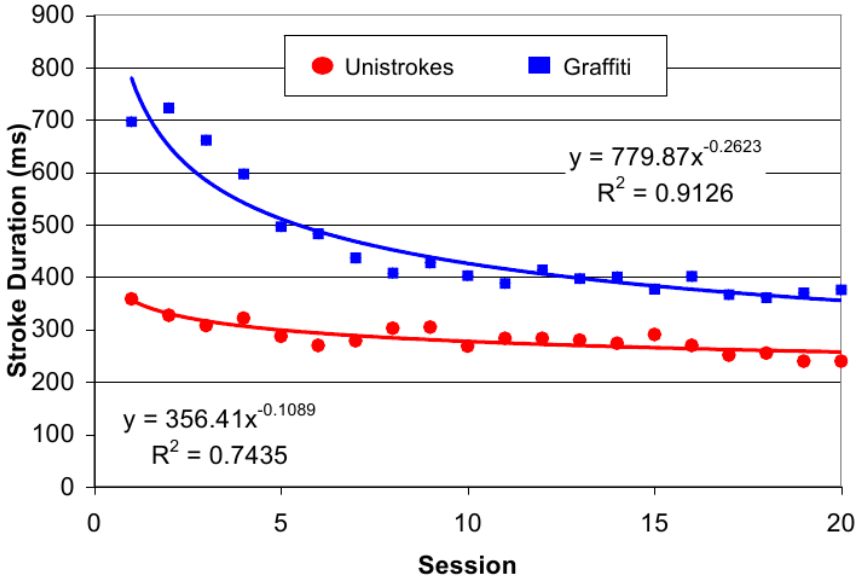
3.7.2 Graffiti

Graffiti on Palmin[2] kehittämä elepohjainen tekstinsyöttömenetelmä. Erityisesti Graffitia käytetään Palmin valmistamissa omissa PalmOS-kämmentietokoneissa. Kuvassa 3.6 esitetyt Graffitin eleaakkoset muistuttavat erittäin paljon roomalaisia kirjaimia. Graffitin eleet ovat Unistrokesin tapaan yksivetoisia.

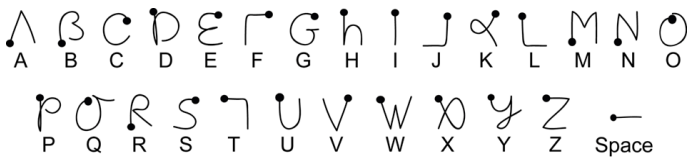
MacKenzie ja Zhang suorittivat Graffitille käytettävyystutkimuksen, jossa he tutkivan menetelmän välitöntä käytettävyyttä (engl. *immediate usability*) [34]. He havaitsivat, että koehenkilöt oppivat Graffitin aakkosista 86% ensimmäisen minuutin aikana ja seuraavan viiden minuutin aikana 97% [34]. Koehenkilöt kykenivät muistamaan tuon kuuden minuutin aikana opetellut eleet vielä viikonkin jälkeen 97% tarkkuudella. He osoittivat, että eleaakkoston ikonisuus ja menetelmän intuitiivisuus ovat erittäin merkittäviä tekijöitä menetelmän oppimisen kannalta alkuvaiheessa.

Költringer ja Grechenig vertailivat Graffitin ja PalmOS:n virtuaalisen QWERTY-

[2]http://www.palm.com



Kuva 3.5: Unistrokes- ja Graffiti-eleiden suoritusajat [12].



Kuva 3.6: Graffiti-eleaakkosto.

näppäimistön käytettävyyttä, virhemääriä ja nopeuksia [28]. Koehenkilöt olivat aloittelijoita, mutta suurin osa kahdestatoista kohenkilöstä oli kokeillut Graffitia muutamia kertoja ennen koetta. Jokainen koehenkilö harjoitteli lisäksi kokeen yhteydessä menetelmää hieman ennen varsinaisia mittauksia. Graffitin kirjoitusnopeudeksi tekstiä kirjoitettaessa mitattiin keskimäärin 9,24 wpm ja kokonaisvirhemääräksi TotalER=19,35%.

Koehenkilöt olivat Graffitilla kirjoittaessaan selkeästi hitaampia kuin kuin virtuaalisella näppäimistöllä [28]. Virtuaalisen näppäimistön kirjoitusnopeudet olivat keskimäärin 13,64 wpm. Koehenkilöt tekivät virtuaalisella näppäimistöllä myös vähemmän virheitä. Kokonaisvirhemäärä oli TotalER = 4,11%. Huonommista tuloksista huolimatta, koehenkilöt pitivät Graffitilla kirjoittamista miellyttävämpänä johtuen sen intuitiivisuudesta ja normaalia käsialaa muistuttavasta aakkostosta.

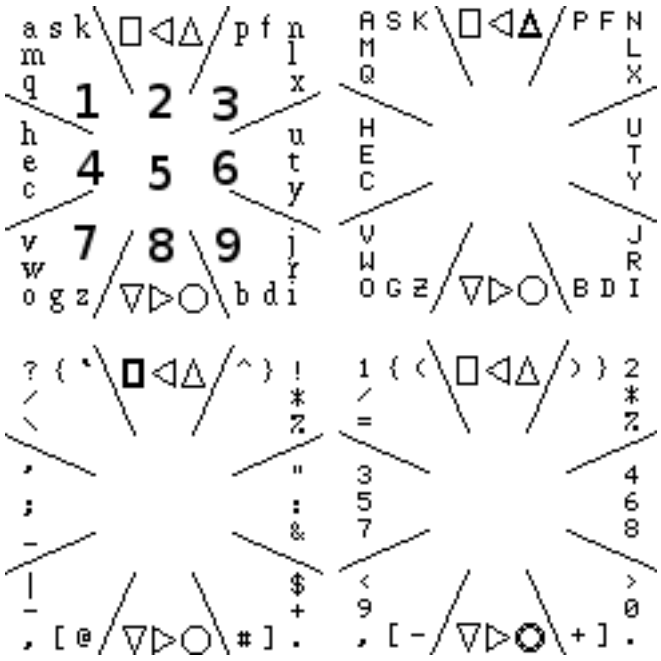
Edellisessä luvussa 3.7.1 esitettyn Castelluccin ja MacKenzien tutkimuksessa Graffitin suorituskyvylle mitattiin samankaltaisia arvoja [12]. Graffitin kirjoitusnopeus ensimmäisellä testikerralla oli 4,0 wpm (σ=1,44) ja viimeisen testikerran jälkeen 11,4 wpm (σ=3,60).

3.7.3 Quikwriting

Quikwriting on Ken Perlinin kehittämä eleisiin perustuva tekstinsyöttömenetelmä [40]. Sen ideana on tuottaa tekstiä elesarjoilla, siten ettei käyttäjän tarvitse nostaa osoitinkynää kosketuspinnalta eleiden välissä. Elesarja voidaan siis mieltää yhtenä jatkuvana eleenä. Menetelmä ei määrittele yksittäisten eleiden hahmoja tarkasti eleaakkoston perusteella, vaan eleet perustuvat osoitinkynän siirtoihin liikealueen täsmällisesti ositettujen ruutujen välillä. Menetelmä on siten sekoitus sekä tunnistus- että valintamenetelmiä.

Quikwritingissa liikealue on jaettu yhdeksään osa-alueeseen kuvan 3.7 esittämällä tavalla. Osa-alueita ovat keskusalue sekä kahdeksan suunta-aluetta. Osa-alueet yksilöidään numeroilla väliltä 1-9, siten että vasemman yläkulman suunta-alueen numero on 1, keskusalueen 5 ja oikean alakulman 9.

Ele tuotetaan siirtämällä osoitinkynä keskusalueelta johonkin kahdeksasta suunta-alueesta ja sieltä joko viereiselle suunta-alueelle tai takaisin keskusalueelle. Osoitinkynän palautus keskusalueelle merkitsee eleen päättymistä. Jokainen ele vastaa yhtä symbolia tai tekstinsyötön kannalta merkittävää operaatiota, kuten väliyöntiä, askelpalautusta tai rivinvaihtoa. Merkistöä vaihdetaan aakkosten, numeroiden sekä erikoismerkkien välillä myös elein. Kuvassa 3.7 välilyöntiä, askelpalautusta sekä rivinvaihtoa vastaavat nuolet oikealle, vasemmalle ja alas vastaavassa järjestyksessä. Merkistö vaihdetaan numeroiksi ympyrää vastaavalla eleellä, erikoismerkeiksi suorakaidetta

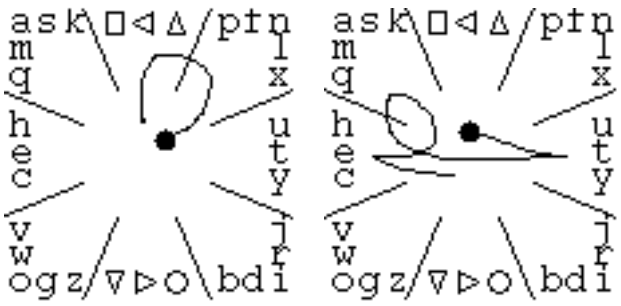


Kuva 3.7: Quikwriting-merkistö.

vastaavalla eleellä ja pienten sekä isojen kirjainten välillä ylöspäin suunnattua nuolta vastaavalla eleellä. Merkistöjä vaihtavat eleet ovat jokaisessa merkistössä samat.

Kuvassa 3.8 on esitettynä eleet yksittäisen merkin ja sana tuottamiseksi. Musta piste kuvastaa paikkaa, jossa osoitinkynä on asetettu kosketuspinnalle.

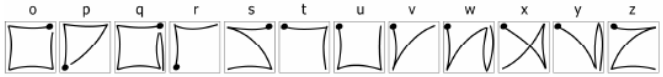
Yksittäiset eleet sulautuvat toisiinsa ja kirjoittaminen on jatkuvaa osoitinkynän liikettä. Quikwriting eroaa tässä suhteessä edellisissä luvuissa esitetyistä Graffitista ja Unistrokesta. Niissä tuotetaan yksittäisiä eleitä, joiden alkupiste on piste, jossa osoitinkynä koskettaa kosketuspintaa ensimmäisen kerran ja loppupiste on piste, jossa osoitinkynä nostetaan pois pinnalta.



(a) Symbolin 'f' kirjoitus. (b) Sanan 'the' kirjoitus.

Kuva 3.8: Quikwriting-kirjoitus.

Kuva 3.9: Palmin kämmentietokone EdgeWrite-kehikolla varustettuna. Kirjoituseleet suoritetaan neliönmuotoisella alueella.



Kuva 3.10: Ote EdgeWrite-eleaakkostosta

Quikwritingissa eleet voidaan käsittää numerosarjoina, jonka alkiona ovat alueiden numerot vierailujärjestyksessä. Sanan 'the' kirjoitus muodostuu siten osoitinkynän vierailuista alueilla 5, 6, 5, 4, 1, 5, 4, 5 vastaavassa järjestyksessä. Quikwritingissa eleentunnistus pelkistyy siten yksinkertaiseen numerosarjojen vertailuun. Toteutus ei tarvitse monimutakisempaa eleiden liikeratojen geometrisia ominaisuuksia tarkastelevaa tunnistusalgoritmia.

3.7.4 EdgeWrite

EdgeWrite on Wobbrockin erityisesti motorisesti rajoittuneita käyttäjiä varten kehittämä kynäeleisiin perustuva tekstinsyöttömenetelmä [59]. Wobbrock havaitsi, että kämmentietokoneiden pienet virtuaaliset näppäimistöt ja tarkkoja liikkeitä vaativat elemenetelmät, kuten Graffiti, ovat motorisesti rajoittuneille vaikeita käyttää. Erikoisryhmien käyttäjät tarvitsevat menetelmän, joka helpottaa hienomotoristen liikkeiden suorittamista.

EdgeWriten perustavana ideana on tehdä eleitä fyysisesti neliönmuotoiseksi rajatulla kosketuspinnalla. Neliönmuotoinen alue voidaan rajata suuremmasta kosketuspinnasta esimerkiksi tarkoitusta varten valmistetulla, kuvassa 3.9 näyvällä, muovikehikolla.

EdgeWriten eleet ovat yksivetoisia liikkeitä neliönmuotoisen liikealueen kulmien välillä. Toisin kuin useimmissa muissa elemenetelmissä, EdgeWriten tunnistusalgoritmi ei

perustu eleen geometristen ominaisuuksien mittaamiseen. Liikealueen kulmat ovat numeroituja, kuten Quikwriten liikealueen osaset. Osoitinkynän vierailu liikealueen kulmissa talletetaan numerosarjoina, jotka yksilöivät muodostetun eleen. Kuvassa 3.10 on esitettynä ote EdgeWriten eleaakkostosta. Kuvan eleistä näkyy selvästi eleiden yksilöinti kulmien perusteella. Jokaisella eleellä on oma yksilöllinen sarja kulmia. Liikealueen rajoittaminen fyysisesti kehikolla rajoittaa myös mahdollisten liikkeiden joukkoa ja tekee siten tunnistusalgoritmista yksinkertaisen.

4 Eleet ja niiden tunnistus

4.1 Eleen määritelmä

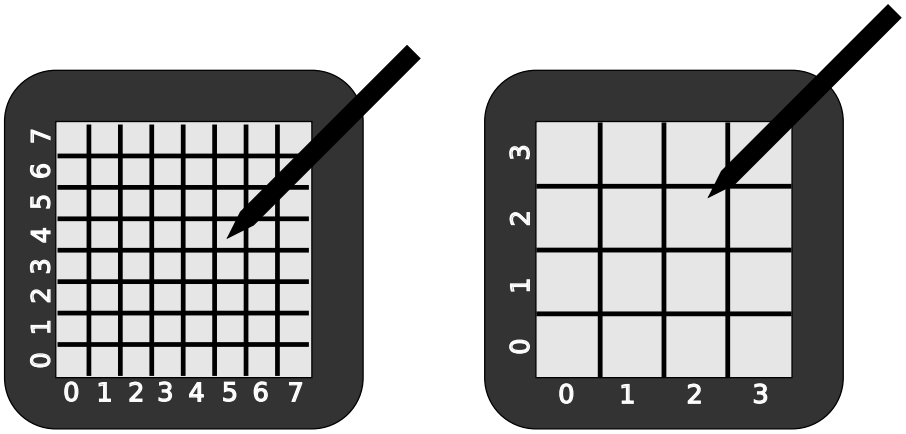
Elehtiminen on yksi inhimillisen kommunikaation esiintymismuodoista. Se voi ilmetä käsien ja vartalon liikkeenä tai ilmeinä. Nespoulos määrittelee eleen olevan **viestimiseen tarkoitettu liike, joka on yleisesti tunnistettu ja jonka merkitys on ilmeinen** [38]. Eleitä käytetään myös tietokoneiden käyttöliittymissä perinteisten ohjausmentelmien, kuten hiiren ja näppäimistön, tukena tai korvikkeena. Käyttöliittymiä voidaan ohjata hyvin erilaisilla eleillä. Eleet voivat olla konenäköön tai muuhun liikkeentunnistustekniikkaan perustuvaa kohteen liikkeiden seuraamista ja tunnistamista kolmiulotteisessa avaruudessa. Esimerkkinä kolmiulotteisesta eletunnistukseta mainittakoon DataGlove. Se on hansikas, jonka asento sekä paikka määritetään magneettikenttään perustuvan paikantimen avulla [62]. Lisäksi DataGloven sorminivelten asennot määritellään sormia pitkin kulkevien valokanavien siirtämän valon voimakkuuden perusteella. Taipuessaan kanavat siirtävät valoa vaimeammin. Käsien paikantaminen kolmiulotteisessa avaruudessa sekä sormien asentojen määrittäminen mahdollistaa tarkasti käsieleiden käytön yhtenä vuorovaikutuskeinona.

Tämän tutkielman puitteissa eleet rajataan tasolla tapahtuviksi liikkeiksi. Eletunnistus on siis pistemäisen kohteen paikantamista ja liikeradan tunnistusta kaksiulotteisessa avaruudessa. Tavanomainen esimerkki on kynän kärjen paikantaminen piirtoalustalla. Tällaisia kynällä tehtäviä liikkeitä kutsutaan usein kynäeleiksi (engl. *pen gestures*). Tutkielman elemääritelmässä ei rajoituta vain kynän liikkeisiin piirtoalustalla vaan yleisemmin pistemäisen kohteen liikkeiden seurantaan kaksiulotteisessa avaruudessa.

Eletunnistuksen kaksi keskeistä vaihetta ovat liikkeen havaitseminen ja sen tunnistaminen joksikin ennalta määrättyjen eleiden joukosta. Osiossa 4.1.1 esitellään formaali esitystapa eleiden liikeratojen kuvaamiseksi. Liikekuvauksen muodostamista käsitellään osiossa 4.2 ja sen tunnistamista osiossa 4.3.

4.1.1 Elekuvaus

Willems ja Niels määrittelevät kynäsyötteen olevan kynän liikeradan pisteiden joukko [55]. Tämän tutkielman kannalta termi kynäsyöte rajoittaa turhaan suoritustekniikan



(a) Piirtoalustan D_8 molempien akselien erottelukyky on 4 bittiä.

(b) Piirtoalustan D_4 molempien akselien erottelukyky on 3 bittiä.

Kuva 4.1: Piirtoalustojen erottelukyvyt eroavat, mutta osoitinkynien sijainnit ovat identtiset

vain pinnalla liikuteltavan kynän liikkeisiin ja siksi jatkossa ele määritellään yleisemmin pistesyötteenä ottamatta kantaa kohteeseen, jonka liikettä pyritään havaitsemaan. Pistesyöte voidaan tuottaa laitteella, jonka liikerata (engl. *trajectory*) T määritellään olevan kronologisesti järjestetty pistejoukko

$$T = \{\sigma_1, \sigma_2, ..., \sigma_N\}, \tag{4.1}$$

missä piste σ_i on x_i- ja y_i-koordinattien sekä ajanhetken t_i muodostama kolmikko

$$\sigma_i = \{x_i, y_i, t_i\} \tag{4.2}$$

siten, että

$$\forall \sigma_i \in T, t_i < t_{i+1}. \tag{4.3}$$

Toisin sanoen, havaintolaite tuottaa aikaleimattuja pisteitä kronologisessä järjestyksessä. Willems ja Niels liittävät jokaiseen paikkatietoon σ_i vielä merkinnän voimasta, jolla kynää on painettu pintaa vasten, mutta tässä tutkielmassa voima jätetään huomiotta yleisemmän lähestymistavan vuoksi.

4.2 Havaitseminen

Pistesyöte kuvaa siis pistemäisen kappaleen liikettä tasolla. Havaintolaitteen erottelukyky (engl. *resolution*), ulkoinen tarkkuus (engl. *accuracy*), sisäinen tarkkuus (engl.

precision) sekä näytteenottotaajuus (engl. *sampling rate, sampling frequency*) vaikuttavat merkittävästi pistesyötteen laatuun. Arkikielessä sisäinen ja ulkoinen tarkkuus usein sekoittuvat. Myös erottelukyky saatetaan mieltää usein virheellisesti tarkkuuden synonyymiksi. Seuraavat aliluvut havainnollistavat näiden neljän ominaisuuden eroja käytännön esimerkkien avulla. Esimerkeissä havaintolaitteina käytetään kahta kuvan 4.1 esittämää piirtoalustaa.

4.2.1 Erottelukyky

Erottelukyky ilmoittaa nimensä mukaisesti laitteen kyvyn erotella mitattavan kohteen tiloja toisistaan. Suuri erottelukyky mahdollistaa pienienkin tilamuutosten havaitsemisen toisistaan. Vastaavasti pienen erottelukyvyn omaavalle havaintolaitteelle pienet muutokset näyttäytyvät identtisinä. Digitaalisten mittausten erottelukyvyn yksikkönä käytetään tavallisimmin bittejä. Bittien määrä kuvaa havaintojoukon suuruutta. Esimerkiksi kahdeksan bitin erottelukyvyllä voidaan erotella $2^8 = 256$ erilaista havaintoa.

Kuva 4.1 näyttää osoitinkynien sijainnit piirtoalustoilla D_8 ja D_4 ajanhetkellä t_1. Piirtoalustojen erottelukyvyt molemmille akseleille x ja y ovat 8 ja 4 yksikköä tai 4 ja 3 bittiä ilmoitettuna vastaavassa järjestyksessä. Siirretään molempien piirtoalustojen osoitinkyniä identtisesti ajanhetkien t_1 ja t_2 välissä siten, että ajanhetkellä t_2 molempien piirtoalustojen osoitinkynät sijaitsevat kuvan 4.2 osoittamalla tavalla. Katkoviivoin piirretty osoitinkynä esittää vanhaa sijaintia ajanhetkellä t_1. Piirtoalustan D_8 havaitsema osoitinkynän sijainti on muuttunut, mutta piirtoalustan D_4 havaitsema sijainti on sama kuin ajanhetkellä t_1. Piirtoalustan D_4 erottelukyky ei ole siis riittävä havaitsemaan luotettavasti näin pientä osoitinkynän sijaintimuutosta.

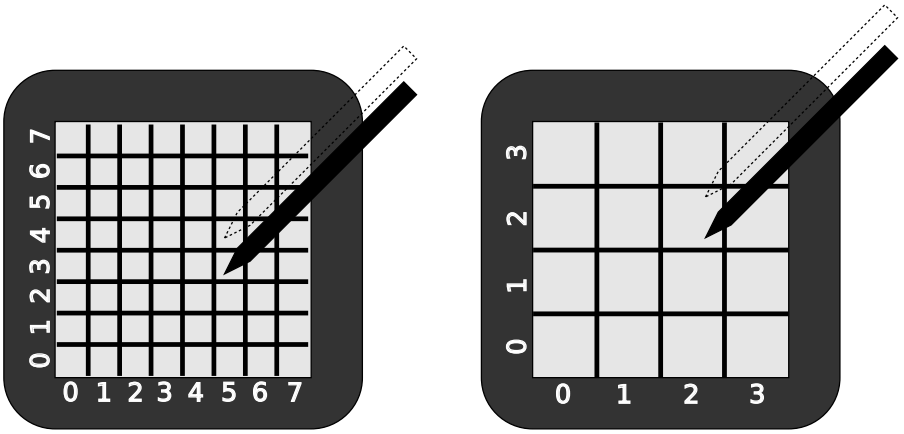
4.2.2 Ulkoinen ja sisäinen tarkkuus

Havaintolaitteen yksi merkittävimmistä ominaisuuksista on sen kyky antaa virheettömiä havaintoja kohteesta. Havainnon kokonaisvirhe E_T on kohteen todellisen tilan σ_{TRUE} ja havainnon σ välinen ero. Kokonaisvirhe koostuu systemaattisesta virheestä E_S ja satunnaisesta virheestä E_R.

Sisäinen tarkkuus kuvastaa havaintolaitteen satunnaisten virheiden suuruutta. Sisäinen tarkkuus on huonoimman yksittäisen havainnon ja havaintojen keskiarvon erotus

$$E_R = \sigma_{WORST} - \sigma_{AVG}. \tag{4.4}$$

Satunnaista virhettä voidaan eliminoida lisäämällä havaintojen lukumäärää, mutta sen luonteenomaisen tuntemattomuuden vuoksi sitä ei voida koskaan täysin eliminoida.



(a) Suuren erottelukyvyn omaava piirtoalusta havaitsee osoitinkynän pienenkin siirron

(b) Pienen erottelukyvyn omaava piirtoalusta ei havaitse pientä siirtymää

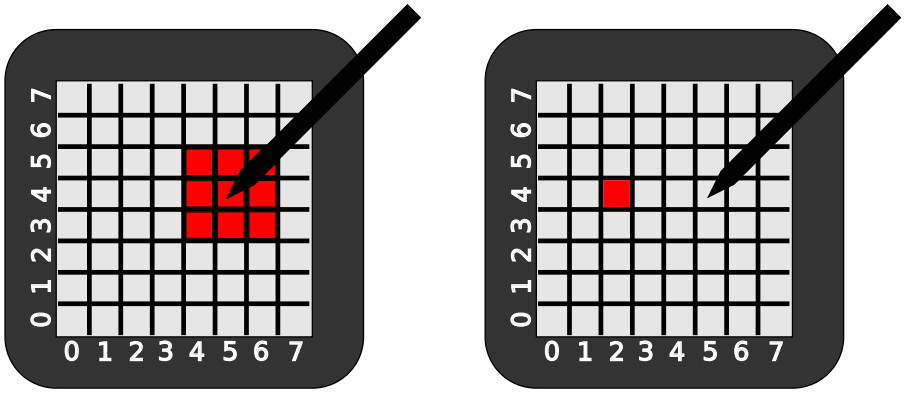
Kuva 4.2: Erottelukyvyn merkitys

Olkoon piirtoalustan D_8 sekä x- että y-akseleiden sisäiset tarkkuudet ± 1 yksikköä ja valitaan piirtoalustalta 9 peräkkäistä havaintoa osoitinkynän pysyessä paikallaan. Tällöin piirtoalusta voi ilmoittaa osoitinkynän sijainneiksi esimerkiksisi kuvan 4.3a punaiset ruudut. Keskiarvoisesti piirtoalustan ilmoittama sijainti on oikea, mutta yksittäinen havainnon mukainen sijainti voi vaihdella merkittävästi todellisesta tilanteesta.

Ulkoinen tarkkuus kuvastaa havaintolaitteen systemaattisten virheiden suuruutta. Se on havaintojen keskiarvon ja todellisen tilan erotus

$$E_S = \sigma_{AVG} - \sigma_{TRUE}. \tag{4.5}$$

Systemaattiselle virheelle on tunnusomaista, että se pysyy havaintokertojen välillä samana tai muuttuu säännönmukaisesti. Systemaattinen virhe voi olla seurausta esimerkiksi epäonnistuneesta havaintolaitteen kalibroinnista. Piirtoalustalla D_8 tällainen kohdistusvirhe voi johtaa esimerkiksi siihen, että havaittujen sijaintien x-komponentit ovat säännöllisesti 3 yksikköä liian pieniä. Kuva 4.3b esittää vastaavan tilanteen. Piirtoalustalta on taas valittu 9 peräkkäistä havaintoa ja jokainen on osoittanut osoitinkynän sijainniksi saman pisteen. Havainnot eivät eroa toisistaan, mutta niiden osoittaman sijainnin ja todellisen sijainnin ero on merkittävä. Toisin kuin satunnaista virhettä, systemaattista virhettä ei voida eliminoida lisäämällä havaintokertoja, mutta se voidaan eliminoida helposti laskemalla havainnolle sopiva korjausmuunnos. Tällaisen korjausmuunnoksen laatimista kutsutaan kalibroinniksi.



(a) Piirtoalustan ulkoinen tarkkuus on hyvä, mutta sisäinen tarkkuus on huono.

(b) Piirtoalustan sisäinen tarkkuus on hyvä, mutta ulkoinen tarkkuus on huono.

Kuva 4.3: Ulkoisen ja sisäisen tarkkuden ero

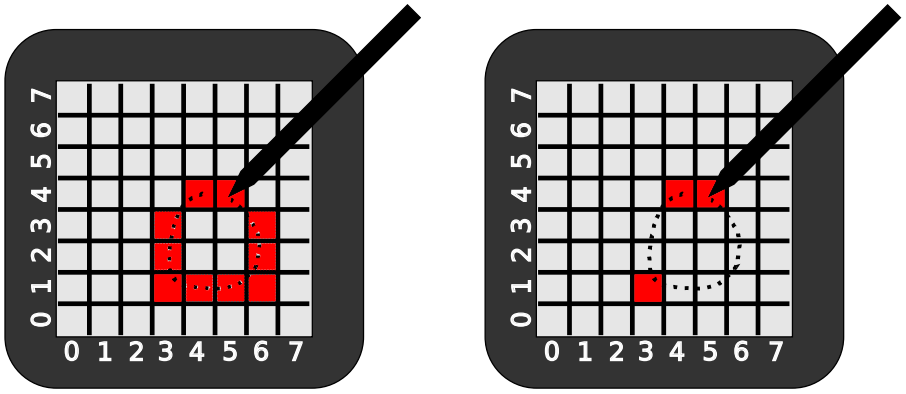
4.2.3 Näytteenottotaajuus

Näytteenottotaajuus ilmaisee laitteen havaintokertojen lukumäärän aikayksikköä, tavallismmin sekuntia, kohden. Näytteenottotaajuutta pienentämällä liikettä kuvaavan pistesyötteen kuvaaja muuttuu suoraviivaisemmaksi ja kulmikkaammaksi. Tällöin nopeiden liikkeiden todellista liikerataa ei havaita. Kuvat 4.4 osoittavat näytteenottotaajuuden merkityksen todellisen liikeradan havaitsemisessa. Näytteenottotaajuuden ollessa riittävän korkea suhteessa seurattavan liikkeen nopeuteen saavutetaan todellista liikettä hyvin kuvastava pistejoukko. Liian matalalla näytteenottotaajuudella havaittu pistejoukko supistuu ja kuvaa todellisesta liikkeestä ei saavuteta.

4.3 Tunnistus

Näppäimistön näppäimet ovat fyysisesti erillään toisistaan. Jokaisen näppäimen painaminen aiheuttaa yksillöllisen signaalin välittymisen näppäimistöltä isäntälaitteelle. Näppäinsignaali ei jätä tulkinnalle sijaa painetun näppäimen identiteetistä. Näppäimistö muodostaa diskreetin äärellisen näppäinavaruuden, jonka jokainen arvo vastaa täsmällisesti jotain näppäintä.

Eleiden tapauksessa tilanne on toinen. Sauvaohjaimilla on mahdollista tuottaa ääretön määrä erilaisia liikkeitä. Eleisiin pohjautuvien menetelmien perusideana on määrittää ennalta jokin määrä eleitä, joihin käyttäjän tuottamia eleitä verrataan. Ennaltamäärättyjen eleiden joukkoa kutsutaan eleaakkostoksi. Tunnistusalgoritmin tehtävä on tunnistaa käyttäjän tuottama ele joksikin eleaakkoston eleeksi. Eleeksi valitaan ta-



(a) Korkealla näytteenottotaajudella saavutetaan todellista liikettä muistuttava kuvio.

(b) Matala näytteenottotaajuus antaa epätodellisen kuvan liikkeestä.

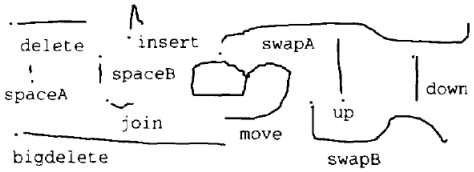
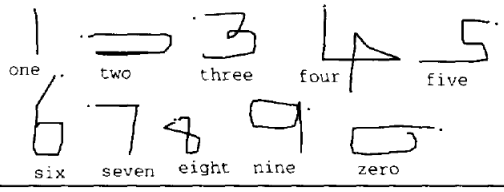
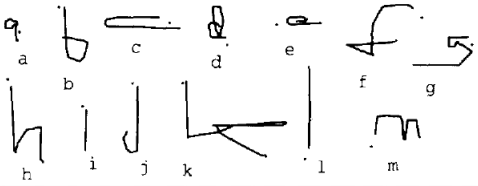
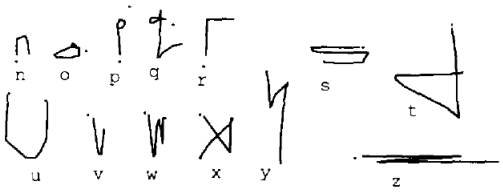
Kuva 4.4: Näytteenottotaajuuden merkitys.

vallisimmin se ele, jota käyttäjän tuottama ele eniten muistuttaa ominaisuuksiltaan.

4.3.1 GRANDMA

GRANDMA on Dean Rubinen kehittämä työkalu tilastolliseen analyysiin perustuvien eletunnistimien luontiin [43]. GRANDMA:lla luodaan eleluokkia määrittelemällä mielivaltainen määrä yksivetoisia mallieleitä edustamaan kutakin luokkaa. Kukin eleluokka edustaa merkitykseltään yhtä toimintoa, tekstinsyöttömenetelmän tapauksessa kirjainta tai muuta kielellistä rakennetta. Kunkin eleluokan ominaisuusvektorin (engl. *feature vector*) tunnusluvuille lasketaan painotukset eleluokan mallieleistä. Tunnusluvun painotus kuvaa sitä, miten hyvin kyseinen tunnusluku kuvaa eleluokan eleitä. Sopivien tunnuslukujen valinta on keskeistä toimivan eletunnistimen luonnissa. Tunnuslukujen pitää pystyä erottamaan eleluokkien eleet toisistaan riittävän suurella todennäköisyydellä.

Tunnistusalgoritmi tunnistaa tuntemattoman eleen kuuluvaksi johonkin määritetyistä eleluokista (C). Tunnistus tapahtuu kahdessa vaiheessa. Ensin eleelle lasketaan vastaava ominaisuusvektori kuin eleluokkia kuvaaville mallieleille. Tämän jälkeen ele luokitellaan kuuluvaksi johonkin eleluokista sen ominaisuuksien perusteella. Luokittelu on yksinkertaisesti sen eleluokan etsimistä, jonka tunnusluvut antavat suurimman arvon tuntemattoman eleen ominaisuusvektorille. Rubine esittää luokittelun formaalisti kaavalla:

Set Name	Gesture Classes	Number of Classes	Recognition Rate
Coleman		11	100.0%
Digits		10	98.5%
Let:a-m		13	99.2%
Let:n-z		13	98.4%
Letters	Union of Let:a-m and Let:n-z	26	97.1%

Kuva 4.5: Tunnistustarkkuuksia erilaisille GRANDMA:lla määritellyille elejoukoille [43].

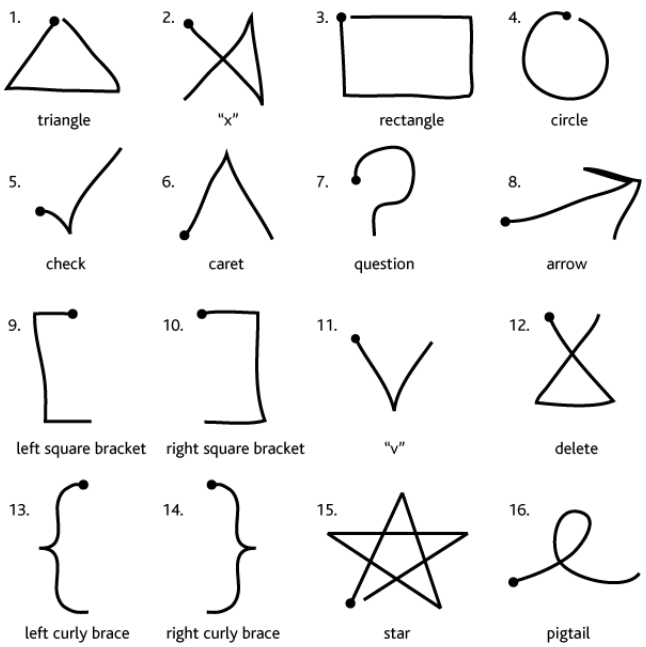
$$v_c = w_{c0} + \sum_{i=1}^{F} w_{ci} f_i, 0 <= c < C \tag{4.6}$$

, missä F on ominaisuuksien joukko, C on eleluokkien joukko, w_{ci} eleluokan c painokerroin ominaisuudelle f_i. Tuntemattoman eleen luokka on se, jolla v_c saa suurimman arvonsa.

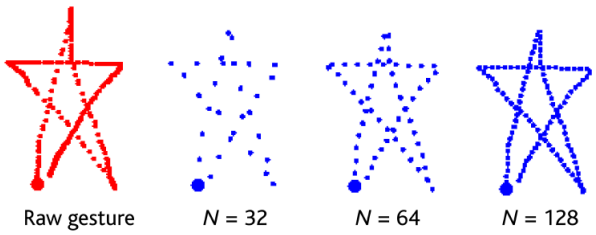
Rubinen ominaisuusvektori koostuu kolmestatoista tunnusluvusta, joita ovat muunmuassa eleen lähtökulma, kokonaispituus ja kiertokulma [43]. Malli mahdollistaa myös helposti uusien tunnuslukujen lisäämisen osaksi ominaisuusvektoria.

4.3.2 $1

Wobbrockin $1-eletunnistusalgoritmi on yksinkertainen, tehokas ja halpa toteuttaa [60]. Se perustuu eleiden ja mallieleiden välisten etäisyyksien vertailuun. Vertailu vaatii



Kuva 4.6: $1-menetelmän arvionnissa käytetty elejoukko [60].



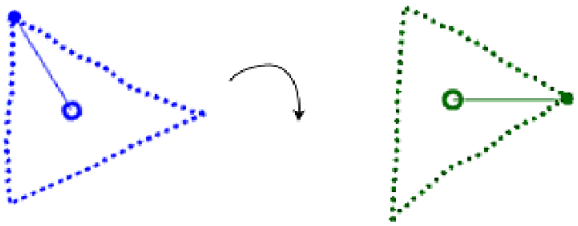
Kuva 4.7: Elekuvauksen yhdenmukaistaminen eri pistemäärille $1-menetelmässä [60].

yksinkertaisten geomoteristen ominaisuuksien tarkastelua ja on siksi helppo sekä nopea toteuttaa. Algoritmin toteutus vie vain noin 100 riviä ohjelmakoodia.

Algoritmi ei vaadi suurta mallielepopulaatiota. Kuvassa 4.6 esitetyn elejoukon tunnistustarkkuus on 97% vain yhdellä mallieleellä jokaista luokkaa kohden [60]. Kolmella mallieleellä tunnistustarkkuus on yli 99%.

Tunnistusalgoritmi koostuu neljästä vaiheesta:

1. Elekuvaus yhdenmukaistetaan vastaamaan mallieleiden elekuvauksia.
2. Elekuvauksen kierto indikaattorikulman perustella.
3. Elekuvauksen skaalaus ja keskittäminen.
4. Elekuvauksen optimaalisen asennon haku.



Kuva 4.8: Elekuvaksen kierto kohti nolla-asentoa [60].

Elekuvaukset voivat sisältää eri määriä pisteitä riippuen laitteen näytteenottotaajudesta ja nopeudesta, jolla ele tehdään. Eleiden ja mallieleiden pisteparietäisyyksien laskemiseksi elekuvausten tulee sisältää aina sama määrä pisteitä. Ensimmäisessä vaiheessa elekuvauksen pistejoukko muokataan tasaväliseksi kuvan 4.7 mukaisesti. Yhdenmukaistus muokkaa elekuvauksen pistejoukon sisältämään tietyn määrän pisteitä siten, että kunkin pisteen etäisyys viereisistä pisteistä on yhtäsuuri.

$1-algoritmi ei erottele eleitä niiden kierroista. Algorimit tunnistaa siis aina neliön neliöksi riippumatta sen asennosta. Toisessa vaiheessa elekuvausta kierretään indikaattorikulman verran kohti nolla-asentoa kuvan 4.8 osoittamalla tavalla.

$1-algoritmi ei myöskään erottele eleitä niiden kokojen tai sijainnin suhteen. Kolmannessa vaiheessa algoritmi skaalaa tunnistettavan eleen mallieleen kokoiseksi ja asettaa elekuvioiden keskipisteet kohdikkain.

Neljännessä vaiheessa suoritetaan varsinainen tunnistus. Tunnistuksessa lasketaan yhdenmukaistetun, kierretyn, skaalatun ja keskitetyn elekuvauksen pisteiden etäisyys suhteessa jokaisen mallieleen elekuvauksen pistettä. Yhdenmukaistuksen johdosta vertailtavat elekuvaukset sisältävät saman määrä pisteitä, joista muodostetaan pisteparit. Elekuvausten vastaavuus on pisteparien etäisyyksien keskiarvo normalisoituna välille $[0, 1]$.

5 Perinteiset peliohjaimet

Tässä luvussa esitellään erilaisista ohjauslaitteista löytyvien sauvaohjaimien toimintaperiaatteita. Luvussa käsitellään sauvaohjaimien toimintaa komponenttitasolla sekä tarkastellaan komponenteilta saatavan datan muuntamista ohjattavan kohteen kannalta merkittävään muotoon. Pääasiallisesti keskitytään useista peliohjaimista löytyvien peukalokäyttöisten sauvaohjaimien teknisiin yksityiskohtiin.

5.1 Sauvaohjaimet

Sauvaohjain (engl. *joystick*) on mekaaninen tai elektromekaaninen ohjauslaite, joka koostuu alustaan kiinnitetystä kiinnityspisteen varassa liikuteltavasta sauvasta. Sauvaohjaimen on tiettävästi keksinyt[1] 1900-luvun alussa ranskalainen lentäjä Robert Esnault-Pelterie lentokoneen ohjainlaitteeksi [47, 50]. Nykyisin sauvaohjaimia käytetään ohjaamaan lentokoneiden lisäksi hyvin suurta joukkoa erilaisia laitteita. Sauvaohjaimia löytyy niin sähköpyörätuoleista, kaivinkoneista kuin peliohjaimistakin. Kulkuneuvojen sauvaohjaimilla ohjataan yleensä niiden liikesuuntaa tai ohjausta vaativia laitteita, kuten kauhakuormaajan kauhaa. Riippumatta käyttökohteesta, sauvaohjaimien tehtävä on muuntaa käden liike ohjattavan kohteen liikkeeksi.

Sauvaohjain voidaan toteuttaa eri tavoin. Alunperin lentokoneen ohjaukseen suunniteltu ohjaussauva oli puhtaasti mekaaninen. Lentokoneissa mekaaninen ohjaussauva on kytkettyinä peräsimen ja siipien vakaimia ohjaaviin vaijereihin. Ohjauskomennot voivat välittyä myös hydraulisesti ohjattaviin kohteisiin. Useille tutuimmat sauvaohjaimet löytyvät kuitenkin peliohjaimista. Niiden elektromekanisten sauvaohjaimien toiminta perustuu sauvan asennon havaitsemiseen elektronisten komponenttien avulla. Tässä tutkielmassa jatkossa sauvaohjamista puhuttaessa tarkoitetaan nimenomaisesti tällaisia elektromekaanisia sauvaohjaimia. Ne jaetaan edelleen niiden toimintaperiaatteiden mukaan digitaalisiin ja analogisiin sauvaohjaimiin, joiden toimintaa esitellään seuraavissa osioissa.

[1] Sekä sauvaohjaimen että sen englannin kielisen terminkin alkuperästä on muutamia ristiriitaisia näkemyksiä. Joidenkin mielestä Yhdysvaltalainen lentäjä ja keksijä James Henry Joyce olisi tämän mullistavan keksinnön takana ja englannin kielinen termi joystick olisi tullut hänen sukunimestään. Toisaalta Oxfordin englannin sanakirjan mukaan kunnia kuuluisi britannialaiselle näyttelijälle ja lentäjälle Robert Lorainelle.

Kuva 5.1: Atari 2600:n peliohjain.

5.1.1 Digitaaliset sauvaohjaimet

Digitaaliset sauvaohjaimet perustuvat nimensä mukaisesti sauvan asennon tai liikkeen täsmälliseen määrittämiseen. Perinteisestä digitaalisesta sauvaohjaimesta esimerkkinä mainittakoon kuvassa 5.1 esitetty, vuonna 1977 julkaistun Atari 2600:n peliohjaimen sauvaohjain. Sauvaohjaimen liikealueen reunoille on asennettuna neljä kytkintä pääilmansuuntien mukaisesti. Sauvaa kääntämällä kohti pääilmansuuntaa vastaava kytkin kytkeytyy ja aiheuttaa signaalin. Väli-ilmansuunnissa sauva aiheuttaa kahden vierekkäisen kytkimen kytkeytymisen. Sauvan ollessa keskellä yksikään kytkin ei ole kytkeytyneenä. Neljällä kytkimellä voidaan siten havaita sauvan yhdeksän eri asentoa.

Sauvan yhdeksän eri asentoa eivät vielä käytännössä mahdollista kohteen ohjaamista. Yhdeksällä asennolla voidaan yksilöidä vain yhdeksän paikkaa kohteen liikeavaruudessa. Esimerkiksi ohjattessa kohdetta neliönmuotoisella tasolla, mikäli vain sauvan asento määrittäisi kohteen sijainnin, muodostuisi kohteelle mahdollinen liikeavaruus kolme yksikköä leveästä ja kolme yksikköä korkeasta ruudukosta. Liikeavaruutta on mahdollista kasvattaa määrittämällä sauvan asento vastaamaan kohteen nopeusvektoria. Sauvan ollessa keskiasennossa, kohteen nopeusvektori on nollavektori. Tarkkailemalla nyt sauvan asentoja säännöllisesti ajan suhteen, voidaan ohjattavan kohteen paikka määrittää triviaalisti vektorilaskennan avulla.

Olkoon funktio v kuvaus sauvan asennosta kohteen nopeusvektoriksi ja määriteltynä yhtälöryhmän 5.1 mukaisesti.

$$\begin{aligned} \alpha(d) &= \frac{d\pi}{4\pi}, d \in [0,7] \\ v(\vec{i}) &= c \begin{cases} (\cos(\alpha(i)), \sin(\alpha(i))) & i \in [0,7] \\ \vec{0} & i == 8 \end{cases}, c \in \mathbb{R}. \end{aligned} \tag{5.1}$$

Nopeusvektorin pituus c määrätään vakioksi siten, että

$$\forall i \|\vec{v(i)}\| = c, c \in \mathbb{R}, i \in [0, 8] \tag{5.2}$$

Ohjattavan kohteen nopeuden suuruus on siis aina sama riippumatta sen suunnasta. Nopeuksien suuruus on toki vain toteutusyksityiskohta. Se voitaisiin hyvin määrätä myös suunnan funktioksi.

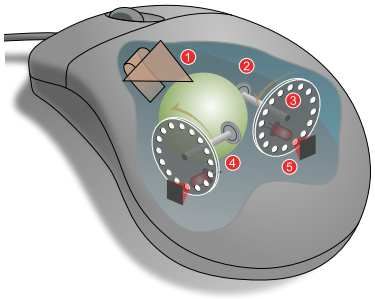
Ohjattavan kohteen paikka ajanhetkellä t määräytyy nyt säännöllisesti tarkkailtavien asentomuutosten sarjana. Jokainen asento kuvautuu omaksi nopeusvektorikseen, joten asentojen aikasarja muodostaa nopeusvektorien summan. Tämä summavektori määrää siten rekursiivisesti kohteen paikkavektorin p suhteessa johonkin mielivaltaiseen ajanhetkeen $t == 0$:

$$\vec{p(t)} = \begin{cases} \vec{0} & t == 0 \\ \vec{v(i)} + \vec{p(t-1)} & t > 0 \end{cases}, t \in \mathbb{N}. \tag{5.3}$$

Ohjattavan kohteen paikkavektori on siis sauvan asennon määrittämän nopeusvektorin ja kohteen edellisen paikkavektorin summa. Edellä esitetyn kaltaiset sauvaohjaimet tunnistavat nimenomaisesti sauvan asennon, mutta eivät sitä liikettä, joka johtaa sauvan lopulliseen havaittavaan asentoon.

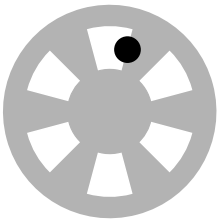
Digitaalinen sauvaohjain voidaan toteuttaa myös ilman kytkimiä mahdollistaen sekä sauvan asennon että liikenopeuden asteittaisen havaitsemisen. Tällaisen konstruktion esitteli hiiren keksijä Doug Engelbart patentissaan [18]. Patentti ei sisältänyt kuvausta sauvaohjaimesta, mutta samaa menetelmää liikkeen digitaalisesta havaitsemista voidaan soveltaa sauvaohjaimeen. Patentin kuvaamassa liikkeentunnistusmekanismissa keskeisessä roolissa ovat toisiaan kohtisuoraan asennetut kaksi kiekkoa, joiden kehälle on tasaisin välimatkoin aseteltuna johtimia. Kiekon vieressä kehän kohdalla on sensori, joka lähetää signaalin johtimen pyörähtäessä sensorin kohdalle. Signaalien lukumäärä vastaa kiekon pyörimää matkaa ja siten hiirellä ohjattavan osoittimen siirtymää. Mitä nopeammin signaaleja välittyy sitä nopeammin osoitin siirtyy.

Kuvassa 5.2 on esitettynä perinteinen pallohiiri, joka perustuu edellä kuvattuun Engelbartin menetelmään. Kuvan hiiressä tunnistus perustuu kiekon toiselle puolelle asennetuun LED-valoon ja kiekon vastakkaiselle puolelle asennettuun fotodiodiin. Fotodiodi on puolijohde, joka synnyttää virtaa valon osuessa sen pinnalle. Se on siis eräänlainen valosensori. Engelbartin menetelmän johtimia vastaavat yksinkertaisesti pienet kiekon kehälle tasavälein tehdyt reiät. Kuvan 5.3 kiekossa on kuusi aukkoa tasaisin välimatkoin toisistaan. Musta piste kuvaa sensoria. Kiekon pyöriessä LED:n säteilemä valo päätyy sensorin pinnalle pulsseittain. Valopulssit synnyttävät virtapulsseja, joiden



1. Pallo pyörii hiiren liikkuessa.
2. Palloa koskettavat pyörät pyörittävät kiekkoja.
3. Kiekkojen kehät on reijitetty valon läpäisemiseksi.
4. Infrapuna-LED:t loistavat reikien läpi.
5. Sensorien havaitsemat valopulssit muunnetaan paikkatiedoksi.

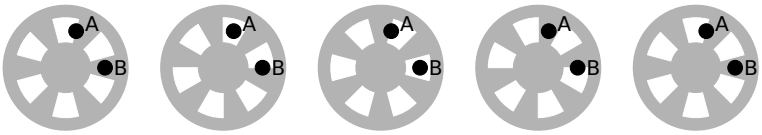
Kuva 5.2: Pallohiiri optisilla sensoreilla.



Kuva 5.3: Reijitetty kiekko ja valosensori.

lukumäärä vastaa hiiren liikkumaa matkaa.

Kiekon pyörimissuunta on mahdotonta määrittää yhdellä sensorilla. Valopulssit näyttäytyvät sensorille samankaltaisina riippumatta siitä katoaako valonsäde sensorista katsoen oikealle tai vasemmalle. Pyörimissuunta voidaan kuitenkin havaita lisäämällä järjestelmään toinen valo-sensori -pari. Sensorit asetetaan havaitsemaan valopulssit hieman eri vaiheissa kuvan 5.4 mukaisesti. Sensori A:n ollessa pimennossa sensori B on juuri aukon kohdalla ja saa hieman valoa. Suunta havaitaan tarkastelemalla sensori A:n viimeisintä havaintoa sensori B:n ollessa pimennossa. Mikäli aukko on siirtymässä A:n kohdalle, kiekko pyörii myötäpäivään. Vastaavasti aukon siirtyessä pois A:n kohdalta, kiekko pyörii vastapäivään.



Kuva 5.4: Kiekon pyörimissuunta havaitaan kahdella sensorilla.

5.1.2 Analogiset sauvaohjaimet

Kuten edellisistä osioista käy ilmi, digitaalinen mittauslaitteisto mittaa kohdetta aina rajatulla arvoalueella. Sen tuottamat arvot ovat aina täsmällisiä, diskreettejä, numeerisia arvoja ja ovat siten vertailtavissa keskenään. Digitaaliset havainnot ovat täsmällisyytensä vuoksi esitettävissä binäärisesti tietokoneen käytettäväksi. Digitaalisuuden vastakohtana voidaan pitää analogisuutta. Liike, kuten kaikki muutkin ilmiöt maailmassa, on luonteeltaan analogista, jatkuvaa. Liike voi suuntautua äärettömään määrään eri suuntia ja edetä äärettömällä määrällä eri nopeuksia.

Useimmat peliohjaimien sauvaohjaimista ovat analogisia. Eräs yleinen tapa mitata sauvan liikettä on havaita sen asentomuutoksia sähkövirran muutoksina potentiometrien avulla. Potentiometri on vastus, jonka resistanssia voidaan muuttaa siirtämällä johdinta vastustavan elementin pinnalla lähemmäksi tai kauemmaksi sisääntulevasta johtimesta. Mitä kauemmaksi ulosvievää johdinta siirretään sitä suuremmaksi resistanssi kasvaa ja sitä vähemmän virtaa potentiometrin läpi johtuu. Johtimen siirtäminen lähemmäksi sisääntulevaa johdinta vähentää vastustavan elementin vaikutusta ja virtaa pääsee enemmän läpi. Potentiometreillä liikettä mittaavan sauvaohjaimen sauva on kiinnitettynä kahden toisiaan kohtisuoraan asennettun aisan hahloihin. Aisat ovat vastaavasti kiinnitettyinä potentiometreihin. Sauvaa kääntäessä aisat pyörivät akseleidensa varassa ja kiertävät potentiometrien johtimia muuttaen läpi johtuvan virran määrää.

5.2 Peliohjaimet

Peliohjaimet ovat kehittyneet Atari 2600:n ajoista merkittävästi. Ensimmäiset peliohjaimet sisälsivät vain sauvaohjaimen ja korkeintaan muutaman painikkeen. Uusimmat peliohjaimet sisältävät useita erilaisia sensoreita, painikkeita ja muita ohjainkomponentteja. Modernit peliohjaimet eivät ole ainoastaan ohjaamista varten. Ne kykenevät myös antamaan aistipalautetta käyttäjän toimintojen pohjalta värinän, valojen tai äänien muodossa. Globaalin tietoverkon syntyminen sekä langattomien tiedonsiirtomenetelmien kehittyminen on tarjonnut aivan uudenlaisia mahdollisuuksia videopelien sisällöntuotantoon. Pelikonsolit kautta maailman ovat reaaliaikasesti yhteydessä toisiinsa.

Langattomat tiedonsiirtomenetelmät mahdollistavat vastaavasti peliohjainten ja niiden ympäristöstä löytyvien laitteiden kommunikoinnin. Pelikonsolien moniydinsuorittimien tarjomat laskentatehot ovat valtavat. Huimat teknologiset kehitysaskeleet ovatkin siirtäneen pelikonsolit videopelien aikakaudesta kohti kotien monipuolisten multimediakeskusten aikakautta. Tämän tutkielman kannalta mielenkiinto on kohdistuneena nimenomaisesti peliohjaimiin ja niiden tarjoamiin tekstinsyöttömahdollisuuksiin. Seuraavissa aliluvuissa esitellään kahden suuren pelikonsolivalmistajan peliohjainvalikoimia ja tekstinsyöttömenetelmiä.

Perinteisellä peliohjaimella tarkoitetaan ohjainta, jonka pääasiallisina ohjainkomponentteina toimivat peukalo-ohjattavat analogiset sauvaohjaimet sekä vaihteleva määrä erilaisia painikkeita. Kuvien 5.7b PlayStation DualShock 3 sekä 5.5b Wii Classic -peliohjaimet ovat hyviä esimerkkejä perinteisistä peliohjaimista. Analogisia sauvaohjaimia käyttäviä peliohjaimia on kahden edellä mainitun mallin lisäksi runsaasti muitakin. Lähes jokaiselta peliohjainvalmistajalta löytyy valikoimastaan ohjaimia, joissa on kaksi peukalo-ohjattavaa sauvaohjainta ja useampia painikkeita.

Perinteisissä peliohjaimissa painikkeet on ryhmiteltynä sauvaohjaimien lähettyville sekä ohjaimen yläreunaan. Yläreunan painikkeita kutsutaan usein olkapääpainikkeiksi (engl. *shoulder buttons*). Perinteiset peliohjaimet tarkoitettu pidettäväksi kiinni kaksin käsin, etusormet ohjaimen yläreunassa ja peukalot sauvaohjaimien päällä. Kämmenet sekä vapaat sormet tukevat ohjainta mahdollistaen peukaloiden sekä etusormien vapaan liikehdinnän. Sauvaohjaimet ja olkapääpainikkeet ovat kaikki kontrolloitavissa samanaikaisesti edellä kuvatulla otteella.

Peukaloiden käyttöä peliohjaimien pääasialliseen ohjaamiseen puoltaa myös myös niiden parempi asentotarkkuus suhteessa muihin sormiin. Refshauge, Kilbreath ja Gandevia tutkivat peukalon asentoaistin tarkkuutta suhteessa muihin sormiin [42]. Tutkimuksessa mitattiin peukalon, etusormen ja nimettömän tarkkuuksia ojennettuina sekä koukistettuina. Tutkimus osoitti, että peukalon tarkkuus suhteessa muihin mitattuihin sormiin oli selvästi parempi koukistettuna. Ojennettuinakin etusormen ja nimettömän tarkkuudet olivat samaa luokkaa kuin koukistetun peukalon. Ei siis lienee sattumaa, että peliohjaimien sauvaohjaimet ovat nimenomaisesti peukaloiden kohdalla eikä esimerkiksi etu- tai muiden sormien kontrolloitavissa.

5.2.1 Nintendo

Nintendon varsinainen pelikonsolihistoria alkoi vuonna 1985 ensimmäisen pelikonsolin Nintendo Entertainment Systemin (NES), julkaisulla [5]. Kuvan 5.5a NESin peliohjain ohjasi Nintendon peliohjainsuunnittelun vuosiksi eteenpäin. Vuonna 1996 julkaistu

(a) NES-pelikonsolin ohjain. (b) Wiimote ja Wii Classic.

Kuva 5.5: Nintendon vanha ja uusi peliohjain.

Nintendo 64:n ohjain toi uutena ominaisuutena yhden peukalolla ohjattavan analogisen sauvaohjaimen sekä mahdollisuuden laajentaa ohjaimen toiminnallisuutta värinällä. Nintendo GameCube:n peliohjain vuodelta 2001 sisälsi kaksi analogista sauvaohjainta. Viimeisien vuosien aikana Nintendo on mullistanut pelikonsolimarkkinoita vuonna 2006 julkaistulla Wii-tuoteperheellään.

Nintendo ei ole julkaissut tarkkoja tietoja Wii-pelikonsolin tai sen ohjaimien toimintaperiaatteista eikä laitetiedoista. Seuraavissa kappaleissa esitetyt tiedot pohajutuvat kirjoittajan omiin havaintoihin sekä Wii-tuotteiden ympärille syntyneen avoimen mutta epävirallisen harrastelijayhteisön yhteisesti julkistamiin tietoihin. Yhteisö koostaa tutkimustiedot WiiBrew-verkkosivustolle[2]. Tiedot laitteista, niiden toiminnasta sekä tiedonsiirtoprotokollista on kerätty pääasiallisesti laitteen toiminnan takaisinmallinnuksella (engl. *reverse engineering*) [7].

Wii-pelikonsolin pääasiallisena ohjaimena käytetään kuvan 5.5b oikeanpuoleista Wii Remote-ohjainta (Wiimote). Ohjaimen komponenttivalikoima on monipuolinen. Se sisältää pienen kameran, kolmiakselisen lineaarikiihtyvyyssensorin, kaiuttimen, neljä LED-valoa, sähkömoottorin värinän tuottamiseksi, kaksitoista painiketta sekä laajennusportin. Ohjain käyttää isäntälaitteen (engl. *host device*) kanssa kommunikointiin Bluetooth-tekniikkaa. Ohjain siirtää tilapäivityksiä isäntälaitteelle 100 Hertzin taajuudella. Isäntälaite voi lähettää ohjauskäskyjä myös ohjaimelle. Tällaisia ohjauskäskyjä ovat esimerkiksi LED-valojen ohjaus, äänentoisto tai värinämoottorin käynnistys ja pysäyttäminen. Ohjain sisältää lisäksi pienen määrän muistia, jota Wii käyttää esimerkiksi Mii-pelaajaprofiilin tallentamiseen.

Wii-pelien keskeinen ohjaustapa on Wiimoten liikkeentunnistus. Ohjaimen päässä

[2]http://wiibrew.org/

	Bits							
Byte	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	RX<4:3>		LX					
1	RX<2:1>		LY					
2	RX<0>	LT<4:3>	RY					
3	LT<2:0>		RT					
4	BDR	BDD	BLT	B-	BH	B+	BRT	1
5	BZL	BB	BY	BA	BX	BZR	BDL	BDU

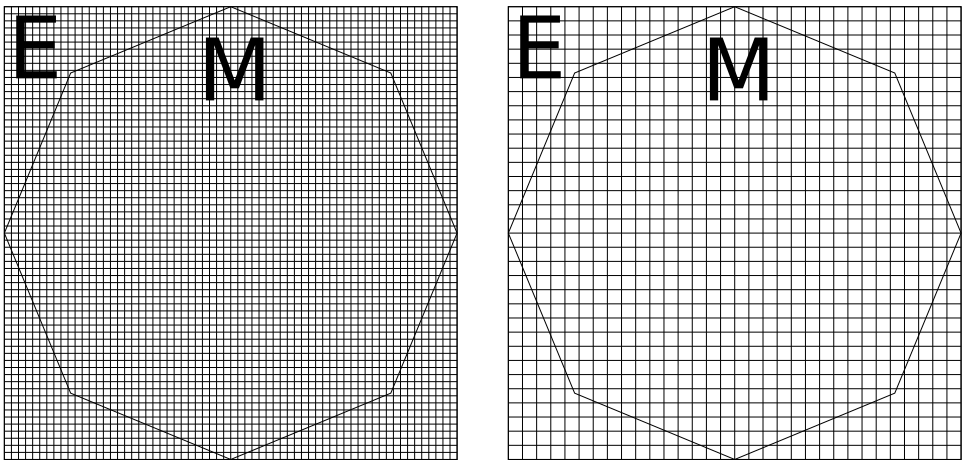
- LX ja LY sekä RX ja RY kuvaavat vasemman ja oikean sauvaohjaimen asentoa vastaavassa järjestyksessä.
- LT ja RT kuvaavat vasemman ja oikean analogisen olkapääpainikkeen tilaa vastaavassa järjestyksessä.
- Kahden viimeisen tavun B-bitit kuvaavat panikkeiden kytkintiloja.

Taulukko 5.1: Wii Classicin tilatietopaketti.

on pieni infrapunavalolle herkkä kamera, jonka kuvainformaatiota käsittelee ohjaimesta löytyvä erillinen laskentayksikkö. Kamera on peitetty mustalla suotimella, joka estää näkyvän valon pääsyn kuvakennolle. Kameran Wii-pelikonsolin varustukseen kuuluu infrapunaledejä sisältävä oheislaite, joka toimii sijaintireferenssinä Wiimoteille. Ohjain havaitsee oman liikkeensä pistemäisten LED-valojen sijaintimuutoksina, jotka ohjain välittää isäntälaitteelle sovellusten hyödynnettäväksi.

Wiimoten lineaarikiihtyvyyssensori mittaa kiihtyvyyksiä liikeavaruuden jokaisen kolmen akselin suhteen. Kiihtyvyyssensorilla voidaan täydentää kameran paikkatietoinformaatiota esimerkiksi mittaamalla ohjaimen asentoa. Ohjaimen asento arvioidaan mittaamalla kiihtyvyyssensorin havaitsemaa kiihtyvyyttä suuhteessa painovoimaan.

Ohjaimen päässä on laajennusportti, johon voidaan liittää lisälaitteita. Kuvan 5.5b ohjaimeen on liitettynä Wii Classic -peliohjain. Wii Classic on nimensä mukaisesti hyvin perinteinen peliohjain. Siinä on viisitoista kytkinpainiketta, joista kaksi olkapääpainiketta ovat myös analogisia. Lisäksi ohjaimessa on kaksi peukaloilla ohjattavaksi tarkoitettua analogista sauvaohjainta. Sauvaohjaimien analoginen data muutetaan digitaaliseksi ohjaimella ja välitetään edelleen Wiimotelle painikeinformaation mukana tilatietopakettina. Taulukossa 5.1 on esitettynä Wii Classicin tilatietopakettien esitysmuoto. Wiimote puolestaan siirtää niin laajennusportin kautta saamansa kuin omien komponenttien datansa isäntälaitteelle.



(a) Vasemman sauvaohjaimen erottelukyky on 6 bittiä.

(b) Oikean sauvaohjaimen erottelukyky: 5 bittiä.

Kuva 5.6: Wii Classicin sauvaohjainten pisteavaruudet (E), liikealueet (M), ja erottelukykyä kuvaava ruudukko. Ruudukkojen tiheydet kuvastavat sauvaohjaimien todellisia erottelukykyjä ja siten myös niiden erottelukykyjen välistä suhdetta.

Wii Classicin sauvaohjaimet ovat komponenteiltaan identtiset. Wii Classicin ilmoittama sauvaohjaindata eroaa kuitenkin erottelukyvyltään. Vasemman sauvaohjaimen akseleiden erottelukyky on 6 bittiä ja oikean 5 bittiä. Erottelukykyjen ero on havainnollistettu kuvassa 5.6. Ruudukko kuvaa sauvaohjaimen sijainnin koko pisteavaruutta sekä erottelukykyä. Jokainen ruutu vastaa yhtä pistettä sauvaohjaimen pisteavaruudessa. Vasen ohjain kykenee erottamaan siis nelinkertaisen määrän pisteitä oikeaan ohjaimeen nähden. Erottelukykyjen ero selittynee Wiimoten rajallisella muistikapasiteetilla. Laajennusporttiin kytketyille laitteille on varattuna vain kuusi tavua muistia tilatietopaketteja varten [7]. Sauvaohjaimien fyysiset liikealueet ovat rajoitettu oktaedrin muotoisella kehyksellä, mikä on havaittavissa kuvasta 5.5b. Fyysisesti rajoitettu alue on koko pistevarauuden alijoukko ja siten kehyksen reunoja pitkin kulkeva liike näyttäytyy oktaedrina sauvaohjaimien asentoja tarkkailevalle sovellukselle.

CWiid-projekti on toteuttanut Wiimoten ja sen lisälaitteiden käyttämät tiedonsiirtoprotokollat C-kielisenä kirjastona WiiBrew-yhteisön laatimien dokumenttien pohjalta [1, 4]. CWiid projektin tuottaman kirjaston nimi on Libcwiid. CWiid-projekti tarjoaa kirjastoon lisäksi Python-kielisidoksen (engl. *language binding*).

(a) PlayStation Control Pad.

(b) PlayStation DualShock 3.

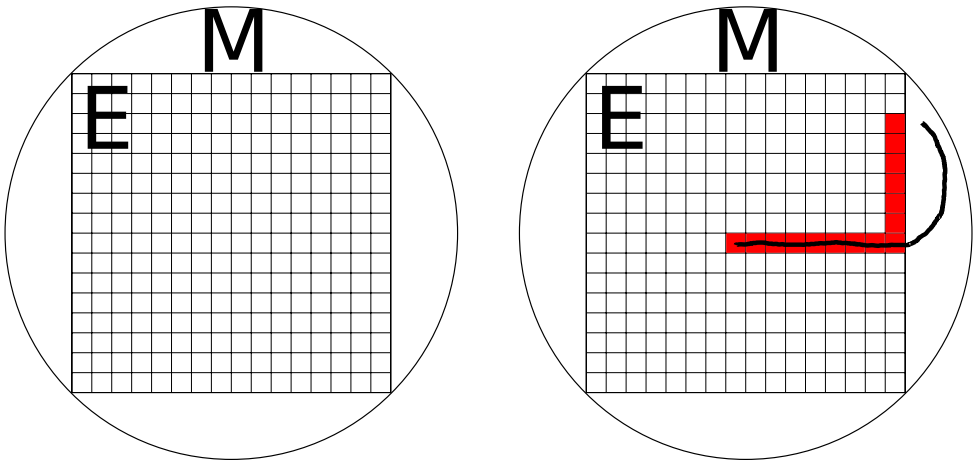
Kuva 5.7: Vanhin ja uusin Sony PlayStation -peliohjain.

5.2.2 Sony

Sonyn pelikonsoliperheen nimi on PlayStation. Ensimmäinen PlayStation julkaistiin vuonna 1994. Sen kuvan 5.7a ohjain muistuttaa paljon Nintendon SuperNintendo-pelikonsolin ohjainta. Molemmissa ohjaimissa on kymmenen painiketta, mutta erottavana tekijänä PlayStation-ohjaimessa on ergonomiaa parantavat kahvat. Sony lisäsi peliohjaimeensa analogiset sauvaohjaimet molempien peukaloiden käytettäväksi kolme vuotta myöhemmin, ehkäpä Nintendo 64:n innoittamana. PlayStation-ohjaimien muotoilu ja ohjainkomponenttivalikoima on säilynyt hyvin pitkälti samanlaisena ensimmäisestä ohjaimesta viimeisiin versioihin saakka. Uusin perinteinen PlayStation-peliohjain on kuvassa 5.7b esitetty DualShock 3. Se sisältää 17 painiketta, joista 12 on analogisia. Tiedonsiirto ohjaimen ja isäntälaitteen välillä on mahdollista joko Bluetooth- tai USB-yhteyden yli. Ohjaimessa on neljä LED-valoa ja kaksi värinämoottoria heikon sekä voimakkaan värinän tuottamiseen. DualShock 3 sisältää Wiimoten tavoin kolmen akselin lineaarikiihtyvyyssensorin.

DualShock 3:n sauvaohjaimien erottelukyky on 8 bittiä. Ne tarjoavat siten Wii Classicin sauvaohjaimiin nähden merkittävästi suuremman erottelukyvyn. Wii Classicin vasempaan sauvaohjaimeenkin nähden DualShock 3:n sauvaohjaimilta on eroteltavissa 16 kertainen määrä pisteitä. Sekä Wii Classicin että DualShock3:n sauvaohjaimien liikealueiden fyysiset mitat ovat kuitenkin identtiset. Niiden halkaisijat ovat $2.2cm$.

Erona Wii Classiciin on lisäksi molempien sauvaohjaimien alla sijaitsevat kytkimet. Käyttäjän on mahdollista painaa niitä sauvaohjaimien normaalin liikuttelun lomassa. Toisin kuin Wii Classicissa, DualShock 3:n sauvaohjaimien ympyränmuotoinen kehys ei rajoita fyysisiä liikealueita. Kehys on suurempi kuin ohjaimen pisteavaruus. Pisteavaruuden ja liikealueen suhde on esitettynä kuvassa 5.8a. Liikealueen pisteavaruutta suuremmista mitoista seuraa, että ympyränmuotoisen kehyksen reunoja pitkin kulkeva liike näyttäytyy loogisesti neliönmuotoisena liikkeenä sauvaohjaimien asentoja tarkkai-



(a) Liikealue on pisteavaruutta suurempi.

(b) Sauvaohjaimen liikeratahavainto vääristyy pisteavaruuden ulkopuolella .

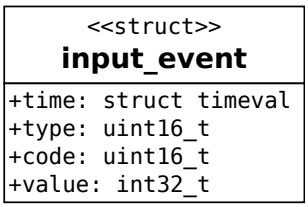
Kuva 5.8: DualShock 3:n sauvaohjaimen piseavaruus (E) ja liikealue (M). Erottelukykyä kuvaavan ruudukon ruudut vastaavat jokainen 16×16 yksikköä, joten todellinen erottelukyky on 8 bittiä molemmille akseleille.

levalle sovellukselle. Tilanne on havainnollistettu kuvassa 5.8b.

DualShock 3 esittelee itsensä USB:n kautta kytkettäessä HID-laitteena. USB HID on USB:n laiteluokka, joka määrittää yleisen rajapinnan erilaisille käyttäjien vuorovaikutuslaitteille. Esimerkiksi peliohjaimet, hiiret ja näppäimistöt kuuluvat HID-luokkaan. USB HID määrittää protokollan, jolla laite ilmoittaa isäntälaitteelle ominaisuutensa. Protokolla mahdollistaa yleisen USB HID-ajurin käytön hyvin erilaisten laitteiden ohjaamiseen ja siten USB HID-standardia noudattavien laitteiden käytön ilman erillisten järjestelmäkohtaisten laiteajureiden asennusta [2].

Linux-järjestelmän USB-väylään kytketty DualShock 3:n tunnistetaan USB HID-laitteeksi ja sen ajuriksi asetetaan geneerinen USB HID-ajuri. Ajuri muuntaa laitteelta tulevan datan syöttölaitetapahtumiksi, jotka Linux-ytimen tapahtumalaitekomponentti Evdev julkaiseen geneerisessä muodossa syöttölaitetiedostona. Sovellukset voivat lukea syöttölaitetiedostosta määrämuotoisina binäärisinä tietorakenteina. Kuvan 5.9 UML-kaavio kuvaa tietorakenteen sisältämät kentät. Kenttien merkitykset ovat seuraavat:

- **item:** Tapahtuman aikaleima ilmaisee tapahtuma-ajan järjestelmän kellon mukaan.
- **type:** Tapahtumatyypillä erotellaan tapahtumakategorioita, kuten näppäimistö- tai akselitapahtumia, toisistaan.



Kuva 5.9: DualShock 3:n syöttölaitetiedostosta luettavan tietorakenteen UML-kuvaus.

- **code:** Tapahtumakoodi yksilöi tapahtumia tyyppien sisällä. Esimerkiksi näppäimistötapahtumien kesken jokaiselle näppäimelle on määriteltynä oma tapahtumakoodi.
- **value:** Tapahtuman arvo ilmaisee tapahtuman luonteen ja sen merkitys määräytyy tapahtumatyypin perusteella. Esimerkiksi näppäimistötapahtumilla mahdollisia arvoja ovat ainoastaan 0 ja 1, jotka ilmaisevat painikkeen vapautuksen ja painamisen vastaavassa järjestyksessä.

Syöttölaitetapahtumia kuvaavaa tietorakennetta käytetään Linux-järjestelmissä kuvaamaan kaikkien erilaisten syöttölaitteiden tapahtumia näppäimistötapahtumista hiirenliikkeisiin. Tapahtumaa kuvaava tietorakenne on määriteltynä kaikissa Linux-järjestelmissä otsaketiedostossa `linux/input.h`. Otsaketiedosto määrittelee lisäksi tapahtumatyyppien ja -koodien sallitut arvot sekä sitoo ne esikääntäjän vakiomakroihin, joilla sovellukset voivat viitata tarvitsemiinsa tapahtumatyyppeihin ja -koodeihin. Esimerkiksi DualShock 3:n sauvaohjaintapahtumien tyyppi on `EV_ABS` ja painiketapahtumien tyyppi `EV_KEY`. Sauvaohjaintapahtumien sisällä tapahtumakoodi erottelee sauvaohjaimen ja akselin. DualShock 3:n vasemman sauvaohjaimen liikuttaminen pystysuunnassa aiheuttaa tapahtuman, jonka tyyppi on `EV_ABS` ja koodi on `ABS_Y`. Tapahtuman arvo määrittää sauvan sijainnin kyseessä olevan akselin suhteen.

6 Pelikonsolien ja peliohjaimien tekstinsyöttömenetelmät

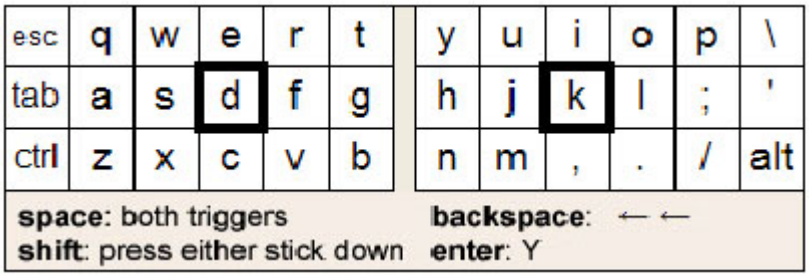
6.1 Fyysiset näppäimistöt

Playstation-pelikonsoleissa tekstiä syötetään pääasiallisesti virtuaalisilla näppäimistöillä tekstinsyöttö on järjestetty Wiin tapaan virtuaalisella valintanäppäimistöllä. Osoitinta liikutetaan D-padilla ja haluttu näppäin valitaan valintapainikkeella. Sony tarjoaa peliohjaimiinsa myös pienen kuvassa 6.1 esitellyn QWERTY-näppämistön lisälaitteena. Se kytkeytyy ohjaimen yläosan USB-portiin. Laite tarjoaa useisiin älypuhelimiin verrattavan pienoiskokoisen näppäimistön käyttömukavuuden kustannuksella. Näppäimistöllä kirjoittaakseen käyttäjän tulee vapauttaa ohjaimesta normaali ote. Kirjoittaminen ja peliohjaimen pääasiallisten ohjauskomponenttien, sauvaohjaimien, käyttö on samanaikaisesti lähes mahdotonta.

6.2 Virtuaaliset näppäimistöt

Nykyisin vallitsevat pelikonsolien tekstinsyöttömenetelmät ova pääasiallisesti erilaset näytöllä navigoitavat valintanäppäimistöt (engl. *selection keyboards*) [27, 56, 58]. Valintanäppäimistöjen toimintaperiaate on yksinkertainen ja yksinkertaisuudestaan johtuen ne ovat myös intuitiviisia. Käyttäjälle näytetään virtuaalinen näppäimistö, jonka näppäinjärjestys noudattaa usein QWERTY- tai aakkosjärjestystä. Kirjain valitaan navigoimalla osoitin toisella sauvaohjaimista halutun kirjaimen kohdalle ja valitsemalla kirjain sopivaa ohjainpainiketta painamalla. Tällaiset vaativaan silmä-käsi -koordinaation

Kuva 6.1: PlayStation DualShock 3 -peliohjain lisävarustenäppäimistöllä.



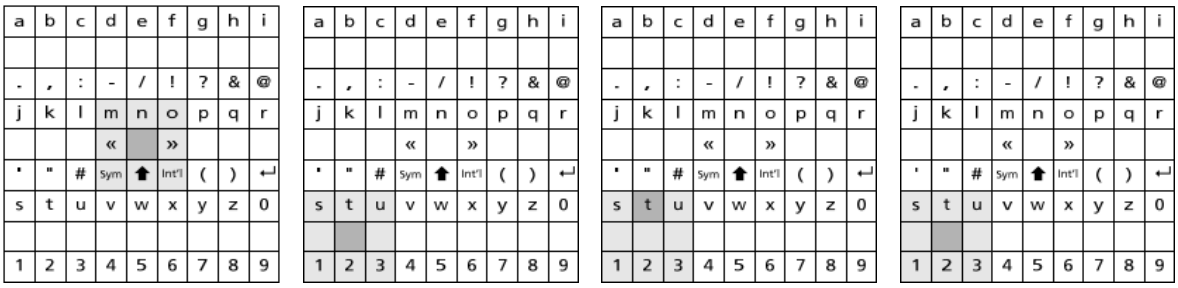
Kuva 6.2: Jaettu QWERTY-näppäimistö ja kaksi osoitinta [56].

perustuvat menetelmät ovat kuitenkin hitaita ja käyttäjien mielestä usein ikävystyttäviä [58]. Valintanäppäimistöt vievät myös suuren osan näyttöpinta-alasta.

Vaihtoehtoiset näppäimistöjärjestyksetkään eivät nopeuta tekstinsyöttöä riittävästi ja vaativat lisäksi käyttäjiä opettelemaan kirjainten uudet sijainnit ennalta tuttuihin näppämistöjärjestyksiin verrattuna [9, 35]. Valintanäppämistöissä peräkkäisten kirjainten väliset etäisyydet ovat suhteellisen pitkiä. Valintanäppämistöissa vuorovaikutetaan myös hyvin usein vain yhdellä ohjainkomponentilla kerrallaan. Tämä johtaa väistämättä siihen, että yhden kirjaimen tuottamiseen käytetään hyvin pitkä aika suhteessa esimerkiksi fyysiseen näppäimistöön oikealla kymmensormitekniikalla. Valintaosoittimen liikenopeutta kasvattamalla nopeuttaa voitaisiin varmasti osaavan käsissä lisätä hieman, mutta vastaavasti osumatarkkuus oikeaan näppäimeen vähenisi vääjämättä.

Wilson ja Agrawala tarjoavat helposti käyttöönotettavaa ja intuitiivista ratkaisua ottamalla käyttöön toisenkin ohjaussauvan ja jakamalla näppämistön kahteen osaan kuvan 6.2 mukaisesti [56]. Heidän menetelmässä vasemmalla sauvaohjaimella navigoidaan näppäimistön vasemmalla puolikkaalla ja oikealla sauvaohjaimella oikealla puolikkaalla. Näppäinvalinnat tehdään vastaavien puolien liipasinpainikkeilla. Kahden osoittimen käyttö lyhentää näppäimistöllä navigoitavaa matkaa.

Heidän tutkimuksensa osoitti, että kahdella ohjaussauvalla kirjoittaminen oli nopeampaa jopa aivan aloittelijoiden joukossa kuin perinteisellä yhden ohjaussasauvan valintanäppämistöllä [56]. Vaikka menetelmä oli aloittelijoidenkin parissa nopeampi kuin täysin perinteinen valintanäppäimistö, kirjoitusnopeudet jäivät kuitenkin vaatimattomaan $7 wpm$:n nopeuteen. Költringer ja Isokoski arvioivat kahden osoittimen käytön tuoman kognitiivisen lisäkuorman mitätöivän liikkeiden lyhenemisen suoman edun [27].



(a) Molemmat sauvaohjaimet keskitettyinä.

(b) Ensimmäinen sauvaohjain vasemmalle alas.

(c) Toinen sauvaohjain suoraan ylös.

(d) Toinen sauvaohjain takaisin keskelle.

Kuva 6.3: Symbolin 't' tuottaminen TwoStickillä. Ensimmäinen sauvaohjain ohjaa vaalean harmaata ruudukkoa ja toinen sauvaohjain ohjaa tumman harmaata ruutua [27].

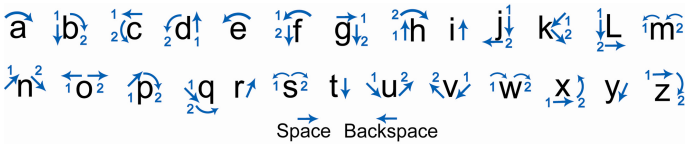
6.3 EdgeWrite sauvaohjaimelle

Wobbrock on jatkokehittänyt luvussa 3.7.4 esitetyn EdgeWriten käytettäväksi myös sauvaohjaimille [58]. Menetelmä käyttää samaa EdgeWriten eleaakkostoa. Wobbrockin suorittamassa käytettävyystestissä, vasta-alkajat kirjoittivat 15 minuutin harjoittelun jälkeen keskiarvoisesti 6.40wpm nopeudella kokonaisvirhemäärän ollessa $TotalER = 10.85\%$. Noin puolituntia pidempään harjoitelleiden kirjoitusnopeus oli 10.43wpm. Nopeimman koehenkilön kirjoitusnopeus oli peräti 12.61wpm, vain vajaan tunnin harjoittelun jälkeen.

6.4 TwoStick

TwoStick on Költringerin ja Isokosken kehittämä tekstinsyöttömenetelmä kahden sauvaohjaimen peliohjaimille [27]. TwoStickin keskeinen ajatus on sauvaohjaimien asentojen jakaminen yhdeksään osaan. Sauvaohjain voi osoittaa johonkin kahdeksasta ilmansuunnasta tai olla lepoasennossa keskellä. Sauvaohjaimen kaikki asennot määritetään vastaamaan 3×3 ruudukon ruutuja. Menetelmän toteuttaminen ei vaadi analogisia sauvaohjaimia. Se voidaan toteuttaa esimerkiksi luvussa 5 esitetyn Atari 2600:n peliohjaimella käytettäväksi. Sauvaohjaimet määritellään loogisessa mielessä toimivan kahdella päällekkäisellä tasolla, ensimäinen sauvaohjainen ylätasolla ja toinen alatasolla. TwoStickin visuaalisesti esittämä kaksitasoinen ruudukko on esitettynä kuvassa 6.3.

Ylätasoa kontrolloivan sauvaohjaimen asento määrittää alatason ruudukon. Alatasolla valinta tehdän viemällä toinen sauvaohjain johonkin kahdeksasta reunaruudusta



Kuva 6.4: UniGest-eleaakkosto [13].

ja palauttamalla se keskelle lepoasentoon. Yhden yksittäisen merkin tuottaminen vaatii siis aina kaksi liikettä. Siten kahden tason mahdollistama suurin yksillölisten valintojen lukumäärä on $9 \times 8 = 72$, koska toisen sauvaohjaimen keskiasento on varattu valinnan määrittämiseen. Isot kirjaimet voidaan tuottaa ohjainpainikkeita käyttämällä.

Költringering ja Isokosken teettämässä käyttäjätestissä käyttäjien kirjoitusnopeudet olivat ensialkuun $5.33wpm$ ($\sigma = 1.33$) ja viiden tunnin ($20 \times 15min$) harjoittelun jälkeen $14.87wpm$ ($\sigma = 3.62$) [27]. Nopein koehenkilö saavutti keskiarvotuloksen $21.24wpm$ viimeisen harjoitusistunnon aikana. Koehenkilöiden kokonaisvirhemäärät olivat $TotalER = 13.34\%$ ($\sigma = 4.93$) ensimmäisen harjoitusistunnon aikana ja $TotalER = 8.21\%$ ($\sigma = 4.43$) viimeisessä harjoitusistunnossa.

6.5 UniGest

UniGest on Castelluccin ja MacKenzien kehittämä tekstinsyöttömenetelmä Wiimotelle [13]. Se perustuu ohjaimella suoritettaviin eleisiin, joiden havaitsemiseen käytetään sekä ohjaimen kameraa että kiihtyvyyssensoria. Kuvan 6.4 eleaakkosto suunniteltiin vartavasten Wiimotea ajatellen, mutta ne perustuvat aiemmin esiteltyyn Unistrokes-eleaakkostoon. Aakkosto sisältää eleet kirjaimille a-z, välilyönnin ja askelpalautuksen. Yhteensä eleaakkosto sisältää 28 elettä.

Eleet tehdään liikuttamalla ohjainta aakkoston nuolien mukaisesti. Ele alkaa painamalla ohjaimen B-painikke pohjaan. B-painikkeen vapatus päättää eleen. Eleen tarkastelu näiden kahden täsmällisesti havaittavien painiketapahtumien välillä auttaa eleiden tunnistuksessa. Mikäli eleitä pyrittäisiin tunnistamaan ohjaimen vapaan liikehdinnän joukosta, tahattomien ja virheellisten eleiden määrä olisi merkittävä. UniGest-aakkoston eleet ovat osin ikonisia. Esimerkiksi l-kirjaimen ele muistuttaa muodoltaan isoa L-kirjainta. Viisi yleistä elettä (e, t, a, o ja i) koostuvat yhdestä liikkeestä ja loput enintään kahdesta liikkeestä. Eleiden liikkeet ovat esitettynä kuvassa 6.4 sinisillä nuolilla. Liikkeitä on kymmenen erilaista: liike jokaiseen ilmansuuntaan ja lisäksi ohjaimen kierto myötä- sekä vastapäivään.

Castellucci ja MacKenzie suorittivat käyttäjätutkimuksen UniGest-liikkeiden suoritusaikojen mittaamiseksi [13]. He havaitsivat että liikkeiden suoritusajat vaihtelivat

keskiarvoisesti 296ms ja 481ms välillä. Hitaimmat liikkeet olivat ohjaimen kierrot ja nopeimmat vasemmalle ja oikealle suuntatuneet suorat liikkeet. He arvioivat liikkeiden suoritusaikojen perusteella menetelmän kirjoitusnopeuden teoreettiseksi maksimiksi 27.9wpm.

6.6 Yhteenveto

Tekstinsyöttömenetelmien kirjo on laaja. Tähän mennessä tässä tutkielmassa on esitelty erilaisia tekstinsyöttömenetelmiä näppäimistöistä eletunnistusmenetelmiin. Menetelmien monimuotoisuus ja suuri lukumäärä kertovat toisaalta tarpeesta syöttää tekstiä mitä erilaisimmissa ympäristöissä ja toisaalta siitä, ettei yksikään olemassa olevista menetelmistä tyydytä kaikkia tekstinsyöttötarpeita yksinään. Työpöytäympäristöissä täysikokoinen näppäimistö on tehokkain tekstinsyöttömenetelmä. Toisaalta taas mobiililaitteissa näppäimistöjen edut ovat jo selvästi pienemmät. Pieni koko tekee kirjoittamisesta vaikeampaa. Kirjoitusnopeus hidastuu ja kirjoitusvirheiden määrä kasvaa. Konekirjoituksesta tuttu kymmensormimenetelmä supistuu kaksi- tai jopa yksisormimenetelmäksi. Pienillä näppäimistöillä on myös vaikeaa harjoittaa katsevapaata kirjoittamista. Käyttäjä joutuu helposti silmäilemään vuoroin näppäimistöä ja vuoroin näyttöä.

Tutkimuksen työn kannalta keskeisessä asemassa ovat peliohjaimille suunnatut tekstinsyöttömenetelmät. Vallitsevat menetelmät ovat pääasiallisesti erilaisia virtuaalisia näppäimistöjä. Niillä kirjoittaminen perustuu näppäimien valitsemiseen ohjaimella ohjattavan osoittimen välityksellä. Virtuaalisilla näppäimistöillä kirjoittaminen on kuitenkin valitettavan hidasta ja käyttäjien mielestä usein epämiellyttävää. Pelikonsolivalmistajat ja sisällöntuottajat ovatkin pyrkineet minimoimaan tekstinsyötön tarpeen tuotteissaan. Siinä missä työpöytäkoneilla pelattavissa peleissä reaaliaikainen pikaviestintä on arkipäivää, pelikonsoleissa jopa nimimerkin kirjoittaminen on työläs operaatio.

Erilaisiin eleisiin perustuvat menetelmät pyrkivät korjaamaan tilannetta. Ne lupailevatkin selvää parannusta virtuaalisiin näppäimistöihin nähden. Aiempien tulosten perusteella näyttäisi, että eleisiin perustuvilla menetelmillä voidaan saavuttaa kohtuullisella harjoittelulla jopa 20 wpm:n kirjoitusnopeus. Se vastaa karkeasti ottaen käsinkirjoituksen nopeutta. Voidaan kuitenkin väittää, että 20 wpm:n kirjoitusnopeus ei vielä riitä vastaamaan modernien tekstinsyöttötarpeiden haasteeseen. Tavanomainen kirjoitusnopeus esimerkiksi täysikokoisella työpöytänäppäimistöllä voidaan arvioida olevan luokkaa 30-60 wpm. Ammattimaiset kirjoittajat saavuttavat yli 100 wpm:n nopeuden. Sähköpostiviestin kirjoittaminen käsinkirjoitusnopeudella ei ole riittävän nopeaa, puhumattakaan pikaviestinnästä pelien tiimellyksessä. Kirjoittajan oma arvio riittävästä

nopeudesta edellä mainittujen tekstinsyöttötarpeiden tyydyttämiseen on 30-40 wpm. Se vastaa tavallisen käyttäjän QWERTY-kirjoitusnopeutta. Mielenkiintoista onkin selvittää, mihin kirjoitusnopeuteen perinteisellä peliohjaimella voidaan päästä.

7 Stipe

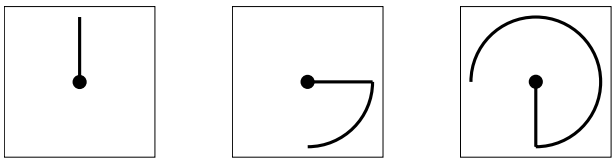
Tässä luvussa esitellään tutkielman kirjoittamista edeltäneen kehitystyön tuloksena syntynyt uusi tekstinsyöttömenetelmä analogisia sauvaohjaimia käyttäville peliohjaimille. Menetelmän nimi Stipe muodostuu englannin kielisestä sanasta *stick* ja englannin kielisen *type*-vebin merkityksettömästä homofonista *tipe*. Stick (suom. *sauva, tikku*) viittaa perinteisten peliohjaimien sauvaohjaimiin. Type tarkoittaa suomeksi näppäilyä, kirjoittamista tai konekirjoittamista. Siten vapaasti ilmaistuna nimi Stipe kuvaa kirjoittamista sauvaohjaimella.

Pelikonsolivalmistajien vallitsevat tekstinsyöttömenetelmät perustuvat lähes poikkeuksetta luvussa 3 esiteltyihin näppäimistöratkaisuihin. Erilaiset näppäimistöt eivät ole optimaalisia ratkaisuja perinteisten peliohjaimien tekstinsyöttömenetelmäksi. Ne ovat hitaita ja hitaudestaan johtuen epämiellyttäviä käyttää. Luvussa 6 esitellyt aiemmat tutkimukset tuovat parannusta ja vaihtoehtoisia menetelmiä, mutta voitaneen rohkeasti todeta, etteivät nekään ole *riittävästi* parempia ratkaisuja.

Alkuperäinen Stipen syntymiseen johtanut idea oli kehittää pelikonsoleille vallitsevia menetelmiä merkittävästi nopeampi tekstinsyöttömenetelmä. Nopeaa tekstinsyöttömenetelmää etsiessä on luonnollista tarkastella näppäimistön ominaisuuksia. Näppäimistön voidaan sanoa olevan laajimmin levinnyt nopean tekstinsyötön mahdollistava menetelmä, joka on sekä intuitiivinen aloittelijoille että tehokas kokeneille. Luvussa 3 kuvattujen näppäimistöjen ominaisuuksista Stipen kehitysperiaatteiksi valittiin tarkkaavaisuustarpeen minimointi ja potentiaalisen tekstinsyöttönopeuden maksimointi.

Stipen kehitystyölle on ollut leimallista erilaisten ideoiden ja toteusvaihtoehtojen kokeileminen. Täysin vastaavanlaista menetelmää ei aiemmin ollut olemassa ja siitä johtuen kehitystyö oli luonteeltaan kokeellista. Tässä luvussa esitellään Stipen elejärjestelmän, algoritmin ja rakenteen lisäksi myös kehitystyön varrella syntyneet ja hylätyt ideat sekä syyt näiden hylkäämiseen. Luku toimii siten koko kehitysprosessia kuvaavana dokumenttina. Kehitystyön aikana hylättyjen ideoiden ja toteutusten dokumentointi on vähintään yhtä tärkeää kuin näennäisesti lopullisen tuotoksenkin esittely. Kehityshistorian tunteminen on perusedellytys onnistuneelle jatkokehitykselle. Jatkokehityksen ohjaaminen oikeaan suuntaan on mahdollista vain tuntemalla tutkimuksen lähtökohdat ja tutkimustyön aikana hylätyt sekä hyväksytyt toteutusratkaisut.

Luvun sisältö on rakennettu seuraavasti. Osiossa 7.1 kuvataan Stipen toiminta yleisellä tasolla. Se antaa lukijalle karkean kuvan kuinka Stipellä tuotetaan tekstiä. Osiossa



(a) Suora ele. (b) Lyhyt kiertoele. (c) Pitkä kiertoele.

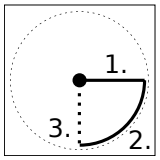
Kuva 7.1: Esimerkkieleet Stipen kolmesta eleluokasta.

7.2 esitetään Stipen nykyinen ohjelmistototetus ja kehitystyössä käytetyt työkalut. Ohjelmistototeuksesta esittellään muun muassa komponenttirakenne, tietovuo sekä tunnistusalgoritmi. Osioissa 7.3-7.7 käydään läpi kunkin komponentin vastuualueet ja toiminta.

7.1 Yleinen toimintakuvaus

Stipellä tuotetaan tekstiä liikuttelemalla peliohjaimen sauvaohjaimia. Sauvaohjaimilla tehdään yksinkertaisia ja lyhyitä liikesarjoja, joita kutsutaan eleiksi. Jokaista elettä vastaa yksi kirjain, joka "syntyy" suorittamalla sitä vastaava ele. Eleet voidaan rinnastaa näppäimistöjen näppäimiin. Näppäimen painaminen vastaa eleen suorittamista. Näppäimistöanalogia sopii myös Stipen tapaan hyödyntää peliohjaimien painikkeita. Niitä käytetään avustamaan tekstinsyöttöä näppimistöistä tuttujuen muuntopainikkeiden, kuten Shift- ja AltGr-painikkeiden, tavoin.

Stipen perustuminen sauvaohjaimiin ja painikkeisiin on luonnollinen valinta olemassa olevan suuren laitevalikoiman vuoksi. Laajassa käytössä olevat peliohjaimet ovat mahdollistaneet Stipen kehityksen täysin ohjelmistopohjaisesti. Kehitystyössä ei ole ollut tarvetta kehittää uutta ohjainlaitetta eikä mukauttaa olemassa olevia laitteita Stipelle sopivaksi. Myös potentiaalisten käyttäjien lukumäärä on suuri. Pelikonsolien lisäksi Stipen voi toteuttaa kaikkiin laitteisiin, joihin mikä tahansa perinteinen peliohjain on kytkettävissä. Tässä tutkielmassa dokumentoitava kehitysversio Stipestä on toteutettuna Linux-järjestelmälle käyttäen luvussa 5 esitettyjä Wii Classic ja DualShock 3 -peliohjaimia.



Kuva 7.2: Eleen liikerata vaiheittain.

1. Sauvan **siirto** liikealueen kehälle suoraan kohti pääilmansuuntaa.
2. Sauvan **kierto** liikealueen kehää pitkin joko myötä- tai vastapäivään.
3. Sauvan **vapautus**, jolloin sauva palautuu liikealueen keskustaan jousivoiman ansiosta.

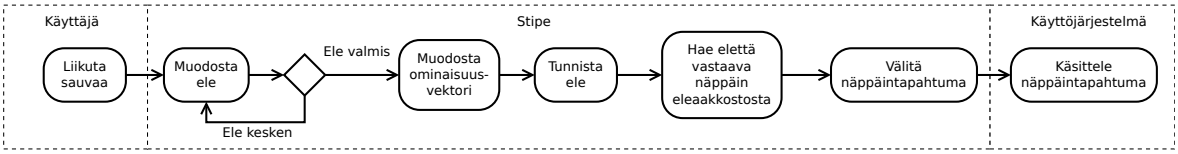
7.1.1 Eleet

Stipen eleet jaetaan kolmeen eri eleluokkaan:

- suorat eleet
- lyhyet kiertoeleet
- pitkät kiertoeleet

Esimerkkieleet kustakin eleluokasta on esitettynä kuvassa 7.1. Kuvassa esitettyjen eleiden notaatio on valittu muistuttamaan Unistrokesia, Graffitia ja EdgeWritea käsittelevien julkaisuiden merkintätapaa yhdenmukaisuuden säilyttämiseksi [12, 19, 59]. Musta piste kuvaa eleen aloituspistettä, josta alkava viiva kuvaa eleen liikerataa. Liikerataa kuvaava viiva päättyy paikassa, jossa käyttäjä vapauttaa otteensa sauvaohjaimesta ja jousi sinkoaa sauvan takaisin keskiasentoon.

Eleet muodostuvat kolmesta eri vaiheesta. Kuvassa 7.2 on esitettynä itään suuntautuvan lyhyen kiertoeleen liikerata vaiheittain. Eleet alkavat aina **siirtämällä** sauva keskiasennosta suoraan kohti pääilmansuuntaa ja päättyvät **vapauttamalla** sauva liikealueen kehällä. Otteen vapauttaminen sauvasta palauttaa sen takaisin keskiasentoon jousivoiman ansiosta. Eleet luokitellaan joko suoriksi eleiksi tai liikealueen kehää kiertäviksi eleiksi. Suora ele tehdään siirtämällä ensin sauva keskiasennosta liikealueen kehälle ja sitten vapauttamalla sauva takaisin keskiasentoon. Kiertoele aloitetaan kuten suora ele, mutta sitä jatketaan **kiertämällä** sauvaa liikealueen kehällä joko myötä- tai vastapäivään. Kiertoeleet luokitellaan edelleen lyhyiksi tai pitkiksi kiertoeleiksi niiden kiertokulman perusteella.



Kuva 7.3: Toimintakaavio Stipen kirjoitusprossessista eleitä tuottamalla.

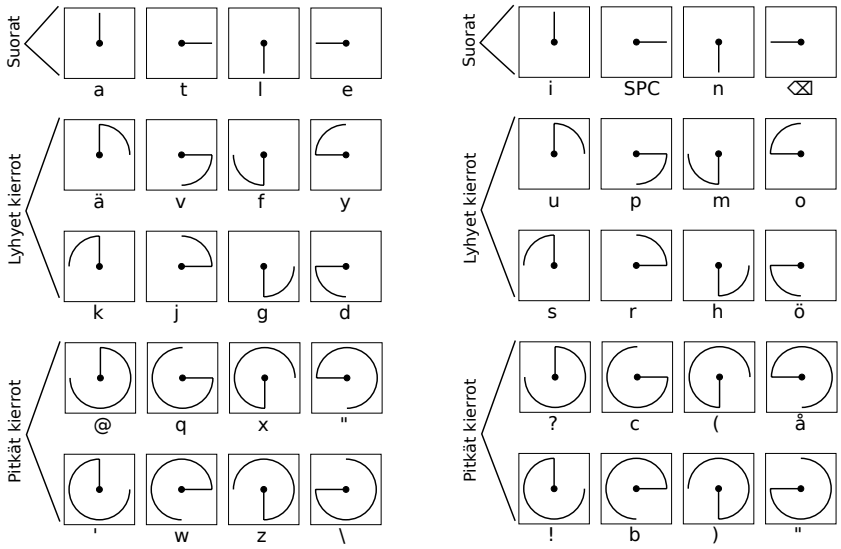
Ele tulkitaan päättyneeksi, kun sauva on palannut takaisin keskiasentoon. Eleen päätyttyä se tunnistetaan joksikin eleaakkoston eleistä ja elettä vastaava näppäintoiminto suoritetaan. Eleaakkoston eleet kuvastavat sauvan ihanteellisia liikeratoja. Todellisuudessa käyttäjien tekemät eleet kuitenkin poikkeavat eleaakkoston eleistä. Sopivimman eleen löytäminen on eletunnistusalgoritmin pääasiallinen tehtävä. Tunnistusalgoritmi saa syötteenä kuvauksen käyttäjän suorittamasta eleestä ja etsii sitä parhaiten vastaavan eleen eleaakkostosta. Edellä kuvattu Stipen kirjoitusprosessi on esitettynä kuvassa 7.3 toimintakaaviona.

7.1.2 Eleaakkosto

Eleitä vastaavat näppäintoiminnot määritellään eleaakkostossa. Stipen tavallisimmat näppäintoiminnot sisältävä eleaakkosto on esitettynä kuvassa 7.4. Eleaakkoston oikean sauvaohjaimen suora ele vasemmalle on askelpalautus ja oikean sauvaohjaimen suora ele oikealle tuottaa välilyönnin. Muut eleet tuottavat niiden kohdalle merkityn symbolin.

Eleaakoston eleiden yksilöllisyys määräytyy kahden ensimmäisen liikevaiheen, siirron ja kierron, perusteella. Kuten kuvasta 7.4 käy ilmi, eleillä on neljä siirtosuuntaa ja viisi kiertokulmaa. Eleiden kiertokulmat ovat suorilla eleillä π, lyhyillä kiertoeleillä $\frac{\pi}{2}$ ja pitkillä kiertoeleillä $\frac{3\pi}{2}$. Kiertokulman negatiiviset arvot kuvaavat kiertoa myötäpäivään ja positiiviset arvot vastapäivään. Kolmas liikevaihe, eli vapautus, on kaikilla eleillä aina samanlainen. Se on jousivoiman tuottava nopea liike kohti liikealueen keskustaa. Eleiden neljästä suunnasta ja viidestä kiertokulmasta seuraa, että sauvaohjaimella voi tehdä 20 yksilöllistä elettä. Täten kaksi sauvaohjainta mahdollistavat yhteensä 40 yksilöllistä elettä.

Stipen eleet eivät ole ikonisia. Ne eivät muistuta muodoiltaan tai liikeradoiltaan niitä vastaavia kirjaimia. Ikonisuuden saavuttaminen eleillä, joiden alkupiste on kiinnitettynä aina samaan pisteeseen osoittautui mahdottomaksi. Sauvat ovat fyysisesti sidottuina liikealueen keskustaan ja siten mielivaltaisten liikeratojen suorittaminen ei ole mahdollista. Kynäelejärjestelmissä eleiden alkupisteet eivät ole sidottuina edellisen eleen loppupisteseen. Kynä voidaan asettaa mielivaltaiseen pisteeseen kosketuspinnalla ja siten myös mielivaltaisten kuvioiden piirtäminen on mahdollista. Stipen eleiden



(a) Vasemman sauvaohjaimen eleet.

(b) Oikean sauvaohjaimen eleet.

Kuva 7.4: Tavanomaisimmat symbolit ja niitä vastaavat eleet listattuna eleaakkostossa.

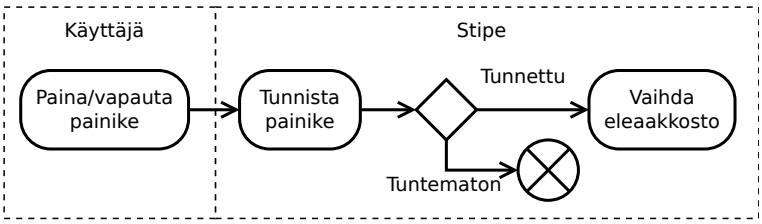
suunnittelun tärkein päämäärä oli luoda joukko helposti suoritettavia, mutta mahdollisimman hyvin toisistaan erottuvia eleitä.

Eleiden suoritusajat kasvavat niiden liikeratojen näennäisen monimutkaisuuden kasvaessa. Suorien eleiden suoritusajat ovat lyhimmät ja pitkien kiertoeleiden suoritusajat ovat vastaavasti pisimmät. Näppäintoimintojen sijoittaminen eleaakkoston eleisiin perustuu suomen yleiskielen kirjainten esiintymistiheyksiin [41]. Yleisimmät kirjaimet ovat sijoitettuna suoritusajoiltaan lyhyisiin eleisiin ja harvinaiset kirjaimet pitkiin eleisiin. Yleisimpien näppäintoimintojen joukkoon lasketaan mukaan myös askelpalautus sekä välilyönti.

Tässä työssä esiteltävä toteutus Stipestä on suunniteltu nimenomaisesti suomen kielen kirjoittamiseen. On kuitenkin ilmeistä, että eleaakkoston merkeiksi voitaisiin yhtä hyvin valita muitakin symboleita kuin länsimaalainen aakkosto täydennettynä skandinaavisilla kirjaimilla. Esimerkkinä mainittakoon liitteessä A.2 esitetyt kehitystyön yhteydessä laaditut eleaakkostot japanin kielen Hiragana-tavuaakkostolle.

7.1.3 Näppäimistöanalogia

Eleet ovat Stipen perustoiminto ja muodostavat Stipe-menetelmän ytimen. Menetelmä käyttää lisäksi peliohjaimien painikkeita hyödykseen. Painikkeita käytetään näppäimistöistä tuttujen muuntonäppäinten (engl. *modifier keys*), kuten Shift- ja AltGr-



Kuva 7.5: Toimintakaavio painikkeiden painamisesta.

näppäinten, tapaan. Näppäimistöjen muuntonäppäinten painaminen muuttaa samanaikaisesti painettavien muiden näppäimien toimintaa. Esimerkiksi Shift-näppämimen painaminen samanaikaisesti a-näppäimen kanssa tuottaa ison A-kirjaimen. Stipessä muuntonäppäimia kutsutaan muuntopainikkeiksi, mutta niiden vaikutus on samankaltainen. Edellä kuvattu painikkeiden yksinkertainen käyttötapaus on esitetty kuvan 7.5 toimintakaaviossa.

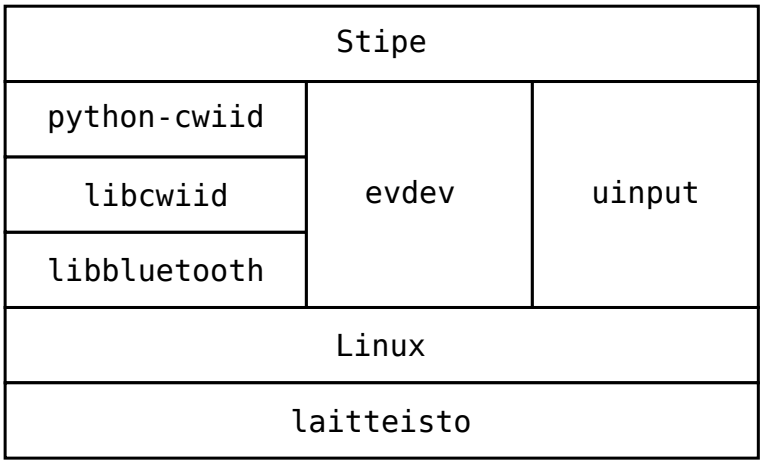
On syytä huomauttaa, että vaikka tässä tutkielmassa Stipestä puhutaan nimenomaisesti tekstinsyöttömenetelmänä, ei menetelmä aseta rajoituksia sen käyttötarkoituksille. Stipen ydinidea on mahdollistaa suuri joukko nopeasti suoritettavia yksilöllisiä toimintoja, joiden merkitys määräytyy käytettävän sovelluksen mukaan. Perusidea on sama kuin näppäimistöissäkin. Näppäimet tai näppäinyhdistelmät vastaaavat tiettyjä symboleita joiden merkitys määräytyy sovellusten tai niiden tilojen mukaan. Tekstinkäsittelyohjelmissa kirjainnäppäimillä syötetään tekstiä, peleissä niillä voidaan ohjata pelihahmoa. Tämän analogian säilyttämiseksi suoraviivaisin toteutus Stipelle on käyttöjärjestelmätason näppäimistöajuri, joka tulkitsee käyttäjän suorittamat peliohjaintoiminnot näppäimistötapahtumiksi.

7.2 Toteutus Linux-käyttöjärjestelmälle

7.2.1 Toteutustekniikat

Stipe toteutettiin Linux-ytimiä käyttäville käyttöjärjestelmille. Kehitystyön tarkoituksena ei ollut tuottaa kuluttajamarkkinoille kaupallisesti valmista tuotetta, vaan ainoastaan luoda prototyyppi konseptin esittelemiseksi, testaamiseksi ja tutkimiseksi. Linux-käyttöjärjestelmän valintaa kehitysalustaksi tukivat sen avoimuus ja sille suunnattujen vapaasti lisenssoitujen työkalujen lisäksi kehittäjän aiempi kokemus kehitystyöstä kyseiselle alustalle.

Stipen lähdekoodi kirjoitettiin enimmäkseen Pythonilla ja pieni osa käyttöjärjestelmäintegraatiota C-kielisenä Python-laajennosmodulina [3, 6]. Python on dynaamisesti



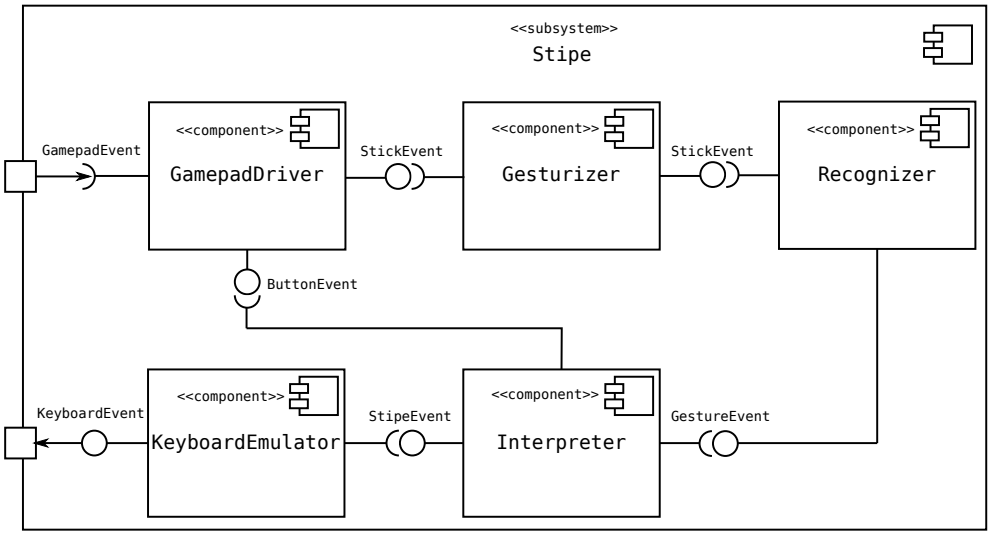
Kuva 7.6: Perinteinen kerroksittainen lohkokaavio Stipen sijoittumisesta suhteessa laitteistoon, Linux-käyttöjärjestelmään ja kirjastoihin.

tyypitetty olio-ohjelmointikieli, joka mahdollistaa muunmuassa korkean abstraktiotasonsa ansiosta nopean tavan tuottaa helposti ylläpidettävää ohjelmakoodia. Python soveltui joustavan luonteensa sekä helpon laajennettavuutensa vuoksi erinomaisesti Stipe-prototyypin kehitykseen. Näppäimistörajapinta käyttöjärjestelmään kirjoitettiin C-kielellä ja muut osat Pythonin versiolla 2.6.

Stipen sijoittuminen suhteessa Linux-järjestelmään, sen tarjoamiin rajapintoihin ja kirjastoihin on esitetty kuvassa 7.6. Kuva esittää yksinkertaistetun kerroksittaisen lohkokaavion, jossa alimpana on laitteisto, jonka Linux-ydin abstrahoi. Ytimen päällä on kirjastopino ja sen laiterajapinnat Evdev sekä Uinput. Kirjastoista Stipen kannalta merkittävimmät ovat CWiid-projektin tuottaman Wiimoten protokollatoteutus libcwiid ja sen Python-kielisidos, johonka pohjautuen Wii Classic -peliohjainajuri toteutettiin. Luvussa 5 kuvattua Evdev-tapahtumarajapintaa käytettiin DualShock 3:n peliohjainajurin toteuttamiseen. Uinput on Linux-ytimen komponentti, joka tarjoaa rajapinnan syöttölaitetapahtumien siirtämiseen käyttäjäavaruudesta (engl. *user space*) ydinavaruuteen (engl. *kernel space*). Se siis mahdollistaa syöttölaiteajurien tai ohjelmallisten syöttölaitteiden integroimisen osaksi Linux-käyttöjärjestelmää ajonaikaisesti käyttäjäavaruudessa ajettavan prosessin toimesta.

7.2.2 Arkkitehtuuri

Stipen komponenttirakenne on esitettynä kuvan 7.7 komponenttikaaviossa. Stipe koostuu viidestä komponentista, jotka muodostavat Stipen sisäisen rakenteen. Stipe itsessään voidaan käsittää alijärjestelmänä, jonka rajapinta ulkopuoliseen järjestelmään on määriteltynä niinikään samassa kaaviossa. Toimiakseen Stipe tarvitsee ulkopuolelta tu-



Kuva 7.7: Stipen komponentit UML-kaaviossa esitettynä.

levia peliohjaintapahtumia ja tarjoaa ulospäin näppäimistötapahtumia. Rajapinta on esitettynä kaaviossa UML-notaation mukaisesti portteina.

Sekä kuvan 7.7 komponenttikaavion että kuvan 7.3 toimintakaavion perusteella Stipen voidaan ajatella olevan funktio, joka kuvaa sauvaohjaimella suoritetun liikkeen käyttöjärjestelmätason näppäintapahtumaksi. Matemaattisessa mielessä funktio on kuvaus lähtöjoukon alkioista maalijoukon alkioiksi. Jokainen lähöjoukon alkio kuvautuu aina samalla tavalla riippumatta kuvaushetkestä tai muista ulkopuolisista tekijöistä. Stipen lähtöjoukko sisältää kaikki erilaiset liikeradat ja maalijoukko näppäistötapahtumat. Sama liikerata tuottaa aina saman näppäimistötapahtuman ja kaikki Stipen näppäimistötapahtumat ovat tuotettavissa määrätyillä liikeradoilla. Toisaalta samaksi näppäintapahtumaksi voi kuvautua useampi eri liikerata. Stipe on siis surjektio.

Stipen funktiomaisuudesta johtuen sen toimintaa voidaan kuvata tunnetuksi arkkitehtuurityyliksi, putkiksi ja suotimiksi (engl. *pipes and filters*) [21]. Putket ja suotimet kuvaavat ohjelmistoa, joka koostuu peräkkäisistä tietovuota käsittelevistä komponenteista eli suotimista. Suotimet on yhdistetty toisiinsa putkilla. Tieto valuu suotimelta toiselle putkia pitkin. Suotimet operoivat vastaanottamaansa tietoa, mahdollisesti muuntavat sitä ja syöttävät sitten eteenpäin seuraavalle suotimelle. Malli kannustaa ohjelmiston kehittämistä modulaariseksi. Mallissa vain peräkkäiset suotimet ovat toisistaan riippuvaisia ja nekin jakavat ainoastaan tiedon niiden välillä liikuvan datan formaatista. Ohjelmiston toimintaa on helppoa muuttaa suodinkomponentteja vaihtamalla. Stipen suodinkomponentit jalostavat peliohjaimen tapahtumia näppäimistötapahtumiksi.

7.3 Peliohjaintapahtumiin reagointi

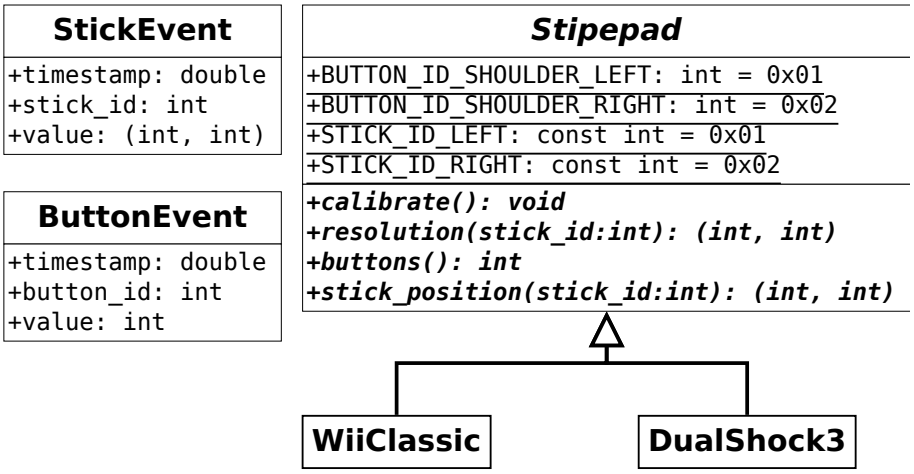
Peliohjainajurin (GamepadDriver) tehtävä on hoitaa tiedonsiirto peliohjaimen ja isäntälaitteen välillä sekä tarjota Stipen komponenteille peliohjainten tilamuutoksia peliohjainriippumattomasti. Peliohjainajurit piilottavat peliohjainkohtaiset eroavaisuudet ja tarjoavat yhtenäisen rajapinnan niitä käyttäville komponenteille. Peliohjainajuria käyttävän komponentin kannalta käytettävällä peliohjaimella ei ole siis merkitystä. Yleisen rajapinnan määrittäminen peliohjainajureille mahdollistaa erilaisten peliohjaimien kytkemisen Stipe-menetelmään ilman, että Stipen muihin ohjelmistokomponentteihin tarvitsee tehdä muutoksia.

Stipe toteutettiin käytettäväksi sekä DualShock 3- että Wii Classic-ohjaimilla. Osiossa 5 esiteltiin näiden kahden peliohjaimen toimintaperiaatteita sekä niiden eroja. Stipen kannalta merkittävimmät erot olivat peliohjaimen ohjaus- ja tilaviestejä välittävä protokolla ja sauvaohjaimien liikealueiden erottelukyvyt. Wii Classic ei myöskään itsessään mahdollista palautevasteiden käyttöä, vaan palutevasteita voi tuottaa ainoastaan Wiimotella. Koska Wii Classicia käytettäessä Wiimote ei ole käyttäjän kädessä, palautevasteiden käyttöä ei tämän ohjaimen kohdalla ole järkevää käyttää. Toisaalta taas DualShock 3 mahdollistaisi teoriassa palautevasteiden tuottamisen sisäänrakennettujen värinämoottorien avulla. Näiden käytön mahdollistavaa protokollaa ei kuitenkaan ole valmiina, joten myös DualShock 3:n tapauksessa palautevasteiden käyttö unohdettiin kehitystyöhön varatun rajallisen ajan vuoksi. Stipen peliohjainajurit toteutettiin siten yksisuuntaisiksi.

7.3.1 Abstrakti peliohjainmalli

Stipe on tarkoitettu yleiseksi tekstinsyöttömenetelmäksi useille perinteisille peliohjaimille. DualShock 3, Wii Classic ja monet muut perinteiset peliohjaimet jakavat keskenään useita ominaisuuksia. Peliohjaimet eivät kuitenkaan ole keskenään täysin yhteensopivia. Kuten luvussa 5 todettiin, niiden ominaisuudet eroavat niin komponenttivalikoiman kuin yhteystekniikankin suhteen. Siinä missä Wii Classicin sauvaohjaimen kiertoliike tuottaa oktaedrin muotoisen kuvion, DualShock 3 tuottaa neliön. Stipen sisäisen toteutuksen kannalta peliohjainmalli on kuitenkin merkityksetön. Totetutuksen ja peliohjainmallin välisen riippumattomuuden takaamiseksi Stipeen kehitettiin hyvin kevyt abstraktiokerros peliohjaimille. Se määrittää yleisen abstraktin peliohjaimen, jonka todelliset toiminnot voidaan toteuttaa erilaisilla peliohjaimilla. Abstraktia peliohjainta kutsutaan vastedes Stipepadiksi.

Stipepadin määrittelemä rajapinta voidaan kuvata kuvassa 7.8 esitetyllä UML-luokkakaaviolla. Kaavion Stipepad on abstrakti luokka. Se esittelee neljä luokka-attribuuttia



Kuva 7.8: Peliohjainajurin luokat.

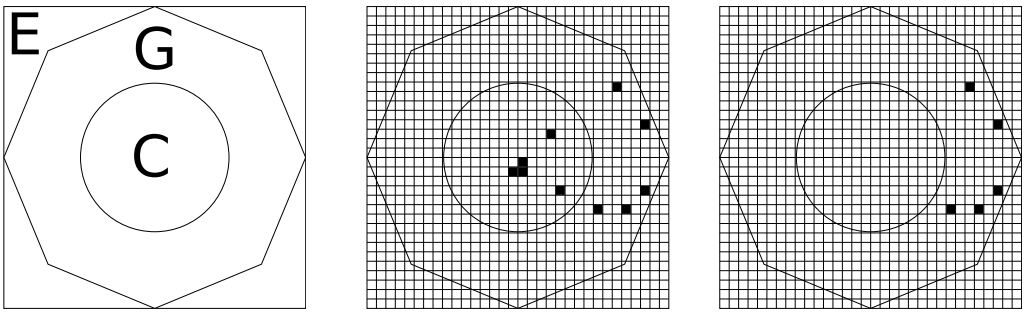
ja neljä abstraktia operaatiota. Luokka-attribuutit antavat yksilölliset tunnusluvut peliohjaimen neljälle ohjainkomponentille. Operaatiot määrittävät ne toiminnot, jotka Stipepad-yhteensopivien peliohjainajurien tulee toteuttaa. Seuraava listaus esittelee operaatioiden toiminnan yleisellä tasolla:

- **calibrate():** asettaa sauvaohjaimien asennon näennäiseksi keskiasennoksi.
- **resolution(stick_id):** palauttaa kokonaislukuparin, joka määrittää sauvaohjaimen erottelukyvyt x- ja y-akseleille.
- **buttons():** palauttaa peliohjaimen painiketiloja kuvaavan kokonaisluvun, joka on painettujen painikkeiden tunnuslukujen kaksikantaisten eksponenttien summa.
- **stick_position(stick_id):** palauttaa kokonaislukuparin, joka kuvaa sauvaohjaimen asennon suhteessa näennäiseen keskiasentoon.

Peliohjainajurikomponentin toteutus koostuu muunmuassa kuvan 7.8 UML-luokkakaavion mukaisista luokista. Stipepad määritää rajapinnan, jonka tiettyä peliohjainta käyttävä ajuri toteuttaa.

7.4 Eleiden muodostus

Peliohjainajuri välittää sauvaohjaimien asentomuutokset elemuodostajalle (Gesturizer). Elemuodostajan tehtävä on suodattaa pois osa peliohjainajurilta vastaanotettavasta



(a) Sauvaohjaimen keskusalueen säde C on neljäsosa koko liikealueen säteestä.

(b) Sauvaohjaimen liikeradan kaikki pisteet.

(c) Eletunnistuksen kannalta merkitykselliset pisteet eli elekuvaus.

Kuva 7.9: Elemuodostaja suodattaa pois sauvaohjaimen liikkeet keskusalueella.

sauvaohjaimen liikkeitä kuvaavasta pistevirrasta ja välittää loput edelleen eletunnistajalle. Eletunnistajalle välitettävää pistejoukkoa kutsutaan elekuvaukseksi.

Peliohjainajurilta vastaanotettava pistesarja virtaa jatkuvasti ja sisältää kaikki sauvan liikkeet. Sauvaohjaimen liikealueen pisteet jaetaan kahteen erilliseen joukkoon, ympyränmuotoiseen keskusalueeseen C ja keskusalueen ulkopuoliseen elealueeseen G. Keskusalue sisältää ne pisteet, joiden euklidinen etäisyys origosta on pienempi kuin neljäsosa akselien erottelukyvystä. Liikealueen jaottelu on esitetty kuvassa 7.9a.

Elekuvauksen ensimmäinen piste on se, jossa sauvaohjain havaitaan ensimmäisen kerran keskusalueelta poistuttuaan. Elekuvauksen muodostaminen päättyy kun sauvaohjain palaa takaisin keskusalueelle. Keskusalue on määritetty kohtuullisen laajaksi sauvaohjaimien keskitinjousien vuoksi. Nopeita eleitä suoritettaessa, sauvaohjain vapautetaan herkästi ääriasennossa aivan elealueen kehällä. Vahva jousi sinkoaa sauvaohjaimen kohti keskustaa, mutta jousivoimasta johtuen sauva voi käydä hyvin nopeasti keskusalueen vastakkaisella sivulla. Pahimmassa tapauksessa sauva osuu nopeasti jousivoiman ansiosta toisen kerran elealueeseen käyttäjän sitä tarkoittamatta. Hyvin pienellä keskusalueella sauvaohjain heilahtelee helposti keskusalueelta pois ja aiheuttaa siten tahattomien eleiden muodostumisen, joita keskusalueen kasvattamisella pyritään suodattamaan pois.

Kohtuullisen laajaa keskusaluetta puoltaa myös sauvaohjaimen kehältä saatava tuntoaistipalaute. Sauvan liike on helpompi havaita, kun sen liikkeestä saa myös tuntoaistimuksia sauvan hiertäessä kehän reunaa. Vastaavia havaintoja tekivät Isokoski ja Raisamo tutkiessaan Quickwriting-menetelmän käyttöä peliohjaimella [23]. He tekivät myös havainnon, että sauvan paluuliike takaisin poikkesi helposti viereisille Quickwriting-sektoreille. Keskusalueen kasvattaminen lyhentää ympyränmuotoisen alueen sektoreita

kapeasta päästä ja siten vähentää sauvan heilahtelun mahdollisuutta viereisten sektorien alueella.

7.5 Eleiden tunnistus

Eletunnistuskomponentin (Recognizer) pääasiallinen tehtävä on etsiä tuntemattomalle eleelle parhaiten sopiva vastine mallipolkujen joukosta. Jokaisella Stipen 20 eleellä on oma mallipolkunsa, joihin eletunnistusalgoritmi tuntemattoman eleen polkua vertaa. Eletunnistuskomponentti välittää tunnistustulokset eteenpäin seuraavalle komponentille tulosten tulkintaa varten. Eletunnistus jaetaan kahteen vaiheeseen:

1. Tuntemattoman eleen ominaisuusvektorin muodostaminen.
2. Täysin sopimattomien mallipolkujen karsiminen.
3. Tuntemattoman eleen polun sovittaminen sopiviin mallipolkuihin.

7.5.1 Ominaisuusvektorin muodostaminen

Elekuvauksesta muodostetaan ominaisuusvektori (engl. *feature vector*). Stipe-eleiden ominaisuusvektori koostuu seuraavista komponenteista:

- Siirtosuunta.
- Kiertokulma.
- Elekuvauksen pisteistä interpoloitu yhtenäinen polku.

7.5.2 Siirtosuunta

Eleen siirtosuunta lasketaan elekuvauksen ensimmäisen pisteen ja pisteavaruuden origon välisen janan suuntakulmana suhteessa x-akseliin.

$$kulma(x, y) = 2 \arctan \frac{y}{\sqrt{x^2 + y^2} + x} \tag{7.1}$$

$$siirtosuunta(elekuvaus) = kulma(elekuvaus_0) \tag{7.2}$$

Suurella keskusalueella ensimmäisen pisteen etäisyys origosta on suuri ja sauvaohjaimen pienet asentoerot keskusalueen ulkopuolella eivät muuta suuntakulmaa merkittävästi. Toisaalta pienellä keskusalueella elekuvauksen ensimmäinen piste on lähellä origoa ja siten pienikin sauvaohjaimen liike aiheuttaa suuren kulmamuutoksen.

7.5.3 Kiertokulma

Kiertokulma lasketaan myös hyvin yksinkertaisesti. Se on origosta elekuvauksen jokaiseen pisteeseen piirrettyjen janojen suuntakulmien summa.

$$kiertokulma(elekuvaus) = \sum_{i=0}^{n} kulma(elekuvaus_i) \tag{7.3}$$

On syytä huomata, että sauvaohjain voi kiertää ympyränmuotoista rataa sekä myötä että vastapäivään. Tästä seuraa, että kulma origon ja mielivaltaisen pisteen välillä voidaan laskea π jaksoissa. Positiiviset arvot kuvaavat kiertoa vastapäivään ja negatiiviset arvot kuvaavat kiertoa myötäpäivään. Elekuvauksen kahden pisteen välinen todellinen kiertoliike ei saisi siten olla puoliympyrää suurempi.

Elekuvauksen pisteiden välinen aikaero määräytyy peliohjaimen päivitystaajuuden perusteella, kuten luvussa 4 kävi ilmi. Sauvaohjaimen liikealueen säde r on $1.1cm$. Keskusalueen säde on puolet liikealueen säteestä. Tällöin lyhimmän mahdollisen puoliympyrän muotoisen kiertoeleen pituus s_{min} on

$$s_{min} = \pi * r * 0.5 \approx 1.73cm \tag{7.4}$$

Peliohjaimen päivitystaajuus on $100Hz$. Elekuvauksen kahden peräkkäisen pisteen välinen aikaero t on siis $10ms$. Jotta kiertokulman ilmoittama kiertosuunta vastaisi sauvaohjaimen todellista kiertosuuntaa, tulisi sauvaohjaimen liikkua havaintojen välillä vähemmän kuin s_{min}. Tästä seuraa, että sauvaohjaimen kiertonopeus v on oltava aina pienempi kuin v_{max}

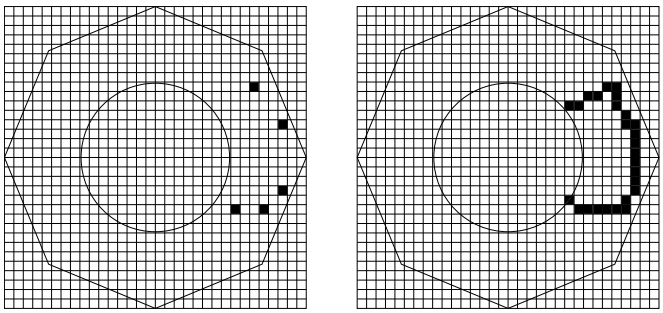
$$v_{max} = \frac{s_{min}}{t} \approx 1.73{}^{m}/_{s} \tag{7.5}$$

$$\forall v, v < v_{max} \tag{7.6}$$

Edellä esitetyn perusteella voitaneen turvallisesti olettaa, että sauvaohjaimia liikuttelevien peukaloiden nopeus ei ikinä saavuta huimaa $1.73{}^{m}/_{s}$ nopeutta ja siten kiertokulmafunktio antaa aina luotettavia tuloksia.

7.5.4 Eleen polku

Elekuvauksen pisteiden lukumäärään vaikuttaa liikkeen nopeus. Sauvaohjaimen päivitystaajuus on vakio, joten nopean liikkeen elekuvaus sisältää vähemmän pisteitä kuin hitaan liikeen elekuvaus olettaen, että liikkeiden liikeradat ovat identtiset. Elekuvaus



(a) Eletunnistuksen kannalta merkitykselliset pisteet.

(b) Bresenham-algoritmin muodostama polku.

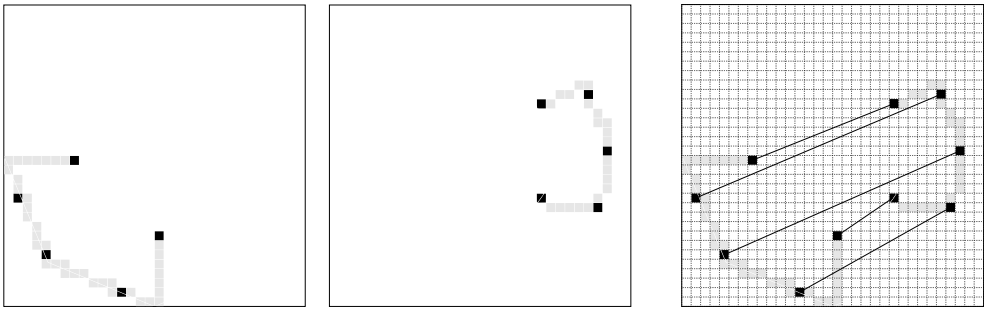
Kuva 7.10: Polun muodostaminen.

ei siis useinkaan muodosta yhtenäistä vieräkkäisten pisteiden sarjaa, vaan elekuvauksen pisteiden väliset etäisyydet vaihtelevat suurestikin. Elekuvausten pistesarjat eivät siten ole suoraan vertailtavissa keskenään. Elekuvausten keskinäisen vertailun mahdollistamiseksi elekuvauksista muodostetaan yhtenäinen polku. Polun muodostamista kutsutaan interpoloinniksi. Pisteiden interpoloimiseen käytetään Bresenhamin algoritmia [10]. Algoritmi antaa pistejoukon, joka muodostaa likimääräisesti suoran viivan kahden pisteen välille bittikartassa. Bresenhamin algoritmi yhdistää siis elekuvauksen pisteet toisiinsa muodostaen yhtenäisen liikerataa kuvaavan polun. Bresenhamin algoritmin Python-toteutus on esitettynä ohjelmalistauksessa B.1. Kuva 7.10 havainnollistaa polun muodostamisen elealueen pisteistä. Polkua muodostettaessa sekä viimeisenä että ensimmäisenä pisteenä käytetään liikealueen origoa.

7.5.5 Mallisovitus

Mallisovitus (engl. *template matching*) on menetelmä, jolla tuntematon ele tunnistetaan sovittamalla eleen polkua mallipolkuihin. Stipen mallisovitusalgoritmi laskee tuntemattoman eleen polun ja mallipolkujen väliset etäisyydet. Polkujen väliset etäisyydet lasketaan tuntemattoman eleen polun ja mallipolun vastinpisteiden (engl. *corresponding points*) välisten euklidisten etäisyyksien summana. Ihannetilanteessa tuntemattoman eleen polku on identtinen mallipolun kanssa, jolloin polkujen vastinpisteiden välisten etäisyyksien summa on nolla.

Algoritmi olettaa vertailtavien polkujen sisältävän saman määrän vastinpisteitä. Vastinpisteparit valitaan siten, että sekä polun viimeinen että ensimmäinen piste tulee vertailtavista poluista valituiksi. Sitten molemmista poluista valitaan tasavälein pisteitä kunnes riittävä pistemäärä on tullut valituksi. Sauvaohjaimen pisteavaruuden suuri



(a) Mallipolun ja tuntemattoman eleen polun vastinpisteet.

(b) Mallipolun ja tuntemattoman eleen polun vastinpisteiden etäisyydet.

Kuva 7.11: Vastinpisteparien valinta ja etäisyydet.

erottelukyky johtaa suuren määrään vastinpistepareja, mikä voi johtaa potentiaalisesti hitaaseen vertailuun. Stipen nykyisessä toteutuksessa pistemääräksi valittiin lyhyemmän polun pisteiden määrä.

Euklidinen etäisyys tuntemattoman polun pisteen p ja mallipolun pisteen t välillä:

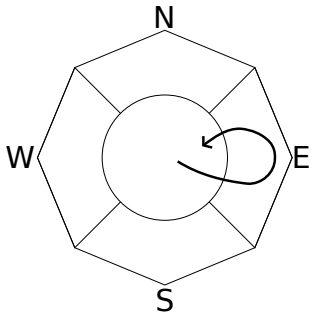
$$distance(p,t) = \sqrt{(p_1 - t_1)^2 + (p_2 - t_2)^2} \tag{7.7}$$

Tuntemattoman polun *path* etäisyys mallipolusta *template*:

$$score(path, template) = \sum_{i}^{N} distance(path_i, template_i) \tag{7.8}$$

Etäisyydet lasketaan tuntemattoman eleen ja *sopivien* mallipolkujen välillä. Mallipolkuja on yhteensä 20. Näistä sopiviksi mallipoluiksi valitaan vain osa kiertokulman ja siirtosuunnan perusteella. Kiertokulman perusteella mallipolkujen joukosta pudotetaan pois ne kiertoeleet, jotka kiertävät päinvastaiseen suuntaan kuin tuntematon ele. Toisin sanoen, myötäpäivään kiertävän eleen polkua ei koskaan soviteta vastapäivään kiertäviin mallipolkuihin.

Mallipolkujen joukosta pudotetaan lisäksi ne polut, joiden siirtosuunta täysin vastakkaisessa suunnassa kuin tuntemattoman eleen siirtosuunta. Koska Stipen mallipolkujen siirtosuunnat ovat pääilmansuunnat, jaetaan elealue neljään sektoriin kuvan 7.12 osoittamalla tavalla. Kuvassa on esitettynä tuntematon ele nuolena liikealueella. Nuoli osoittaa eleen kiertosuunnan olevan *vastapäivään*. Eleen siirtosuunta osuu *itäiseen* sektoriin. Siirtosuunnan perusteella mallipoluista karsitaan pois siksi kaikki *läntiseen* sektoriin osuvat mallipolut. Lisäksi mallipoluista karsitaan pois kaikki *myötäpäivään*



Kuva 7.12: Vastapäivään kiertävä ele, jonka siirtosuunta osuu itäiseen sektoriin.

kiertävät kiertoeleet. Kuva 7.13 havainnollistaa mallipolkujen karsimista kuvan 7.12 eleen perusteella. Mallipoluista jää karsinnan jälkeen jäljelle yhdeksän sopivaa mallielettä. Karsinnan pääasiallinen tarkoitus on vähentää mallisovituksen aiheuttamaa laskennallista kuormaa.

Kun mallipolkujen joukosta on karsittu kiertokulman ja siirtosuunnan perusteella pois sopimattomat mallipolut, lasketaan tuntemattoman eleen polun ja jäljelle jääneiden mallipolkujen väliset etäisyydet. Eletunnistuskomponentti muodostaa eletapahtuman ja asettaa sen kenttien arvot. Eletapahtuman rakenne on esitetty kuvan 7.14 UML-luokkakaaviossa. Etäisyydet mallipolkuihin talletetaan kenttään `scores` järjestettynä kasvavaan järjestykseen. Jokaisella mallipolulla on yksilöivä tunnus. Mallipolkuja vastaava eletunnukset asetetaan kenttään `gesture_ids` etäisyyksiä vastaavaan järjestykseen. Eletunnistaja välittää eletapahtuman tulkille laskettuaan eletapahtuman kenttien arvot.

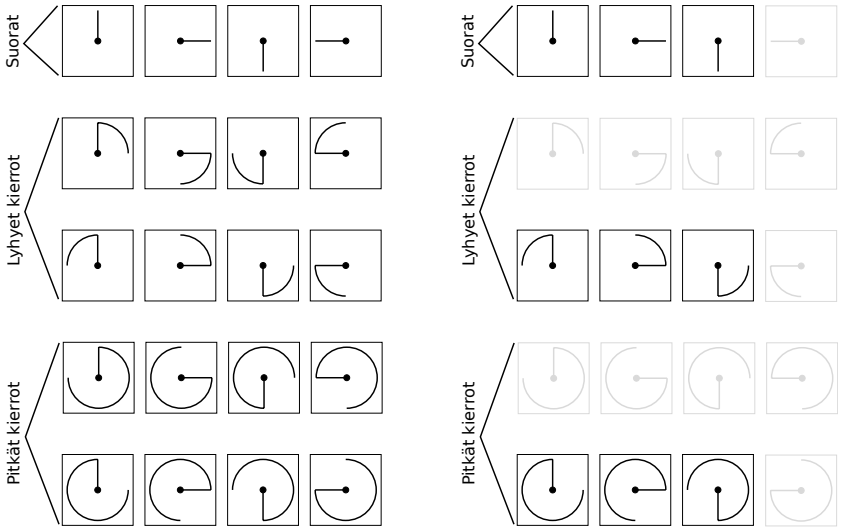
7.6 Eleiden tulkinta

Tapahtumatulkki (Interpreter) vastaanottaa eletapahtumia eletunnistajalta ja peliohjaimen painiketapahtumia suoraan peliohjainajurilta. Tapahtumatulkin tehtävä on päättää, mitä toimenpiteitä havaitut eleet ja painiketapahtumat aiheuttavat.

7.6.1 Eletapahtumat

Eletapahtuman saapuminen eletunnistajalta ilmaisee, että käyttäjä on suorittanut sauvaohjaimella eleeksi tulkittavan liikkeen.

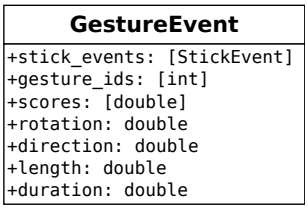
Eletapahtuman `gesture_ids`-kenttä sisältää listan eletunnuksia järjestettynä tunnistustuloksen perusteella kasvavaan järjestykseen. Listan ensimmäinen eletunnus on parhaimmin elettä vastaavan mallipolun eletunnus ja viimeinen huonoimmin elettä vastaavan mallipolun eletunnus.



(a) Kaikki 20 mallipolkua.

(b) Mallipoluista karsittu myötäpäivään kiertävät polut sekä länteen siirtyvät polut.

Kuva 7.13: Mallipolkujen karsinta itään suuntautuvat vastapäivään kiertävän eleen perusteella.



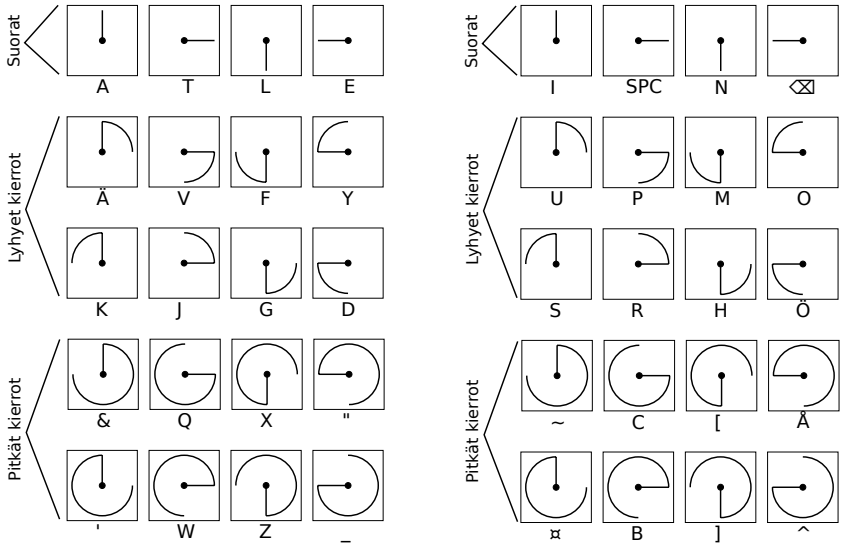
Kuva 7.14: Eletunnistajan tuottamia eletapahtumia kuvaava UML-luokkaesitys.

Tapahtumatulkki valitsee eletunnuslistasta ensimmäisen eletunnuksen ja hakee sitä vastaavan näppäimistötapahtuman eleaakkostosta. Tulkki haetun välittää näppäimistötapahtuman näppäimistöemulaattorin suoritettavaksi.

7.6.2 Painiketapahtumat

Painiketapahtumat aiheuttavat eleaakkoston vaihtamisen tapahtuman muuntopainikekoodia vastaavaksi. Stipen tärkeimmät painikkeet ovat kaksi muuntopainiketta, joiden kytkintilat määrittävät käytettävän eleaakkoston. Yksittäisen kytkimen tila voi olla joko päällä tai pois. Kytkimen tila on siis binäärisesti esitettynä 1 tai 0. Painiketilojen yhteismäärä on 2^n, jossa n on painikkeiden lukumäärä. Kahdella muuntopainikkeella mahdollistetaan neljä erillistä tilaa. Koska muuntopainikkeiden tilat määräävät käytettävän eleaakkoston, kahdella muuntopainikkeella mahdollistetaan neljän erillisen eleaakkoston käyttö. Jokainen eleaakkosto puolestaan sisältää 20 yksilöllistä elettä sauvaohjainta kohti, joten kahdella painikkeella sekä kahdella sauvaohjaimella mahdollistetaan enimmillään 160 yksilöllistä elettä.

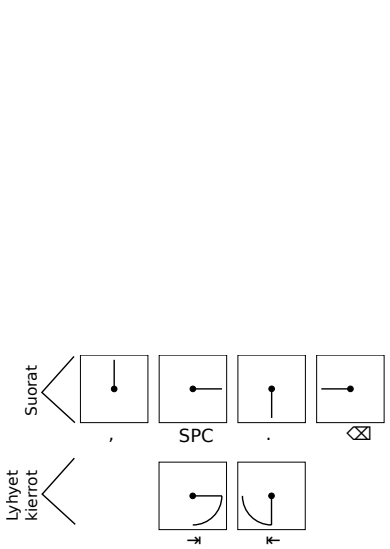
Muuntopainikkeen painaminen muuttaa eleaakkostoa ja vapauttaminen palauttaa eleaakkostoon ennalleen. Stipe käyttää kahta muuntopainiketta. Peliohjaimien olkapääpainikkeet ovat luonnollisin valinta muuntopainikkeiksi. Niiden ohjaaminen etusormilla mahdollistaa sauvaohjaimien liikkuttelun samanaikaisesti peukaloilla. Näin jokaista neljää ohjainkomponenttia voidaan käyttää samanaikaisesti.



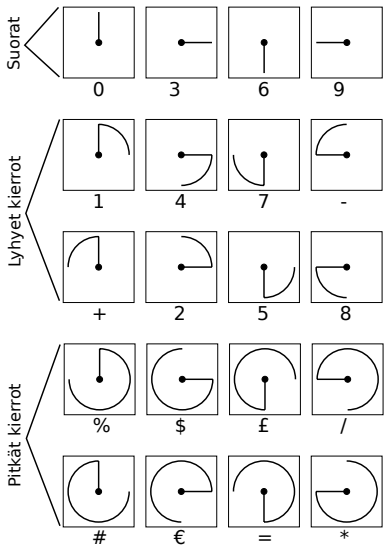
(a) Vasemman sauvaohjaimen isot kirjaimet.

(b) Oikean sauvaohjaimen isot kirjaimet.

Kuva 7.15: Stipen isojen kirjaimien eleaakkosto.



(a) Vasemman sauvaohjaimen erikoismerkit.



(b) Oikean sauvaohjaimen numerot ja erikoismerkit.

Kuva 7.16: Erikoismerkit ja numerot.

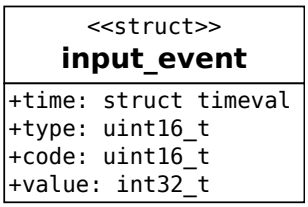
Vasemman olkapääpainikkeen painaminen muuttaa eleaakkoston kuvan 7.15 mukaiseksi. Erona perustilan eleaakkostoon on isot kirjaimet pienten kirjaimien tilalla sekä muuttuneet erikoismerkit. Numeroista ja matemaattisista operaattoreista koostuva kuvassa 7.16 esitetty eleaakkosto valitaan painamalla oikean olkapääpainike.

Numeroita vastaavat eleet valittiin kellotauluanalogiaan perustuen. Numeroita 3, 6 ja 9 vastaavat suorat eleet itään, etelään ja länteen vastaavassa järjestyksessä ilmoitettuna. Numeroa 0 vastaa suora ele pohjoiseen. Numero 1 on yhden **isompi** kuin 0, joten sitä vastaa suoraan pohjoiseen lähtevä **myötäpäivään** kiertävä lyhyt kiertoele. Vastaavasti numero 2 on yhden **pienempi** kuin 3, joten sitä vastaa suoraan itään lähtevä **vastapäivään** kiertävä lyhyt kiertoele. Numerot 4, 5, 7, 8 järjestyvät kuvitteelliseen kellotauluun vastaavalla logiikalla.

7.6.3 Sauvaohjaimen käyttö korvikkeena painikkeille

Stipen kehityksen alkutapaileella ei ollut vielä varmaa onko peliohjaimista löytyvien painikkeiden käyttö järkevää. Painikkeiden käytön ajateltiin rajaavan menetelmän liiaksi vain peliohjaimiin. Painikkeiden käyttöä pohdittaessa ajateltiin, että jos muuntopainikkeiden toiminnallisuus voitaisiin toteuttaa ilman varsinaisia painikketa, mahdollisten Stipeä käyttävien laitteiden joukko voisi olla suurempi.

Muuntopainikkeille tehtiin vaihtoehtoinen toteutus, joka perustui toisen sauvaoh-



Kuva 7.17: Input-alijärjestelmän syöttölaitetapahtumaa kuvaava tietorakenne

jaimen asentoihin eleiden suorittamisen yhteydessä. Menetelmästä käytetään nimitystä nojaus. Nojauksessa toinen sauvaohjain siirretään keskiasennosta kohti yhtä neljästä pääilmansuunnasta. Tällöin toinen sauvaohjain on vapaana suorittamaan eleitä. Nojaavan sauvaohjaimen ilmansuunta määrittää vapaan sauvaohjaimen eleaakkoston. Menetelmän itsensä kannalta on merkityksetöntä kumpaa kahdesta sauvaohjaimesta nojaukseen käytetään. Jos esimerkiksi vasenta sauvaohjainta käytetään nojaukseen, niin oikea sauvaohjain jää vapaaksi eleiden suorittamiseen. Vasemman sauvaohjaimen nojaus pohjoiseen muuntaa oikean sauvaohjaimen eleaakkoston kuvan 7.16a numeroiksi ja nojaus etelään kuvan 7.16b erikoismerkeiksi.

7.7 Näppäimistötapahtumien tuottaminen

Näppäimistöemulaattori (KeyboardEmulator) on komponentti, joka muuntaa Stipen sisäisen esityksen näppäimistötapahtumista käyttöjärjestelmän näppäimistötapahtumiksi. Emulaattori tarjoaa siis Stipen muille komponenteille käyttöjärjestelmäriippumattoman rajapinnan näppäimistötoimintojen tuottamiseen. Komponentin sisäinen toteutus huolehtii Stipen sisäisten näppäimistötapahtumien muuntamisen käyttöjärjestelmän vastaaviksi tapahtumiksi. Käyttöjärjestelmäriippumattomuus mahdollistaa Stipen muiden komponenttien uudelleenkäytön käyttöjärjestelmää vaihdettaessa. Näppäimistöemulaattoria käyttävien komponenttien kannalta käyttöjärjestelmän näppäintapahtumarajapinta ei ole merkityksellinen.

Stipen prototyyppiin kirjoitettiin näppäimistöemulaattori Linux-ytimelle. Linux-ydin tarjoaa C-kielisen Uinput-syöttölaiterajapinnan, jonka avulla sovellukset voivat luoda ajonaikaisesti näppäimistölaitteita osaksi ydintä. Stipen näppäimistöemulaattoria varten kehitettiin erikseen Uinput-rajapinnalle Python-kielisidos, joka mahdollistaa ohjelmallisten syöttölaitteiden luomisen Pythonilla [1].

Uinput on aiemmin kuvatun Evdev-rajapinnan tavoin osa Linux Input-alijärjestelmää

[1] Python-uinput on sittemin Stipestä täysin irroitettu erillinen projekti, joka tarjoaa Python-kielisidoksen Uinput-rajapinnalle. Projektin on julkaistuna GPL-lisenssillä osoitteessa: http://codegrove.org/projects/python-uinput

ja käyttää syöttölaitetapahtumien esittämiseen samaa laitetiedostoon kirjoitettavaa tietorakennetta. Esimerkiksi näppäintapahtumat a-näppäimen painamiselle luodaan kirjoittamalla Uinput-laitetiedostoon tapahtumarakenne, jonka tyyppi on `EV_KEY`, koodi on `KEY_A` ja arvo on 0. Painamistapahtuma samalle painikkeelle on muuten identtinen, mutta arvo on 1.

8 Käyttäjätestit ja menetelmän arviointi

Stipe-menetelmälle suoritettiin sen kehitystyön aikana käyttäjätestejä. Aiemmissa tekstinsyöttömenetelmiä käsittelevissä tutkimuksissa ei ollut juurikaan tehty pitkäaikaista käyttäjätestausta. Kuten luvussa 3 todettiin, kirjoitusnopeus on yksi tärkeimmistä tekstinsyöttömenetelmän laadun mittareista. Käyttäjätesteissä keskityttiin kirjoitusnopeuden kehittymiseen harjoitusjaksojen aikana. Stipe-menetelmän kohdalla koettiin erityisen tärkeäksi tutkia menetelmän kirjoitusnopeuspontentiaalia, joka olisi saavutettavissa pitkän harjoitteluajan jälkeen. Kirjoitusnopeuspotentiaalin ajateltiin olevan uuden tekstinsyöttömenetelmän kohdalla tärkeämpi mittari kuin alkuvaiheen oppimisnopeuden. Mikäli uusi menetelmä on vain vähän olemassa olevia teksinsyöttömenetelmiä nopeampi, sen opettelua on vaikea perustella. Toisaalta, mikäli tekstinsyöttömenetelmän potentiaalinen kirjoitusnopeus on suuri, käyttäjät voivat olla kiinnostuneita uhramaan aikaa myös sen opetteluun. Kysymys on opetteluun vaadittavasta uhrauksesta suhteessa saavutettavaan hyötyyn.

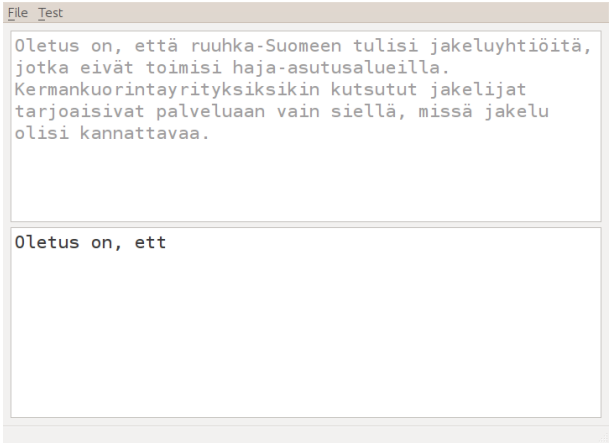
Tutkimuksessa suoritettiin kolme käyttäjätestiä. Käyttäjätestien pääasiallisena tarkoituksena oli kerätä tietoa tekstinsyöttömenetelmän oppimiskäyrästä ja erityisesti kokeneiden koehenkilöiden kirjoitusnopeudesta. Käyttäjätestit sijoittuivat ajallisesti peräkkäin ja niiden koehenkilöt valikoituivat käyttäjätestistä toiseen. Viimeisen käyttäjätestin koehenkilöt kävivät siis läpi myös kaksi ensimmäistä käyttäjätestiä.

Tässä luvussa esitellään ensin käyttäjätestit koejärjestelyineen ja tuloksineen. Luku 8.1 aloittaa käyttäjätestien ja niiden keskeisten tulosten esittelyn ensimmäisestä käyttäjätestistä edeten käyttäjätesteittäin ajallisesti eteenpäin.

8.1 Ensimmäinen käyttäjätesti

8.1.1 Koehenkilöt

Ensimmäisessä käyttäjätestissä menetelmää testattiin kahdellatoista koehenkilöllä. Koehenkilöistä seitsemän oli naisia. Yhtä koehenkilöä lukuunottamatta kaikki olivat alle 30 vuotiaita. Nuorin koehenkilö oli 21 vuotta ja vanhin 42 vuotta. Koehenkilöiden aiempi kokemus peliohjaimista vaihteli jonkin verran. Osa koehenkilöistä harrasti pelaamista silloin tällöin, osa vain satunnaisesti. Peliohjaimet eivät olleet kenellekään kuitenkaan täysin vieraita. Yhdelläkään koehenkilöllä ei ollut diagnosoitua lukihäiriötä tai muuta



Kuva 8.1: Kuvakaappaus testiohjelmasta.

oppimisrajoitetta.

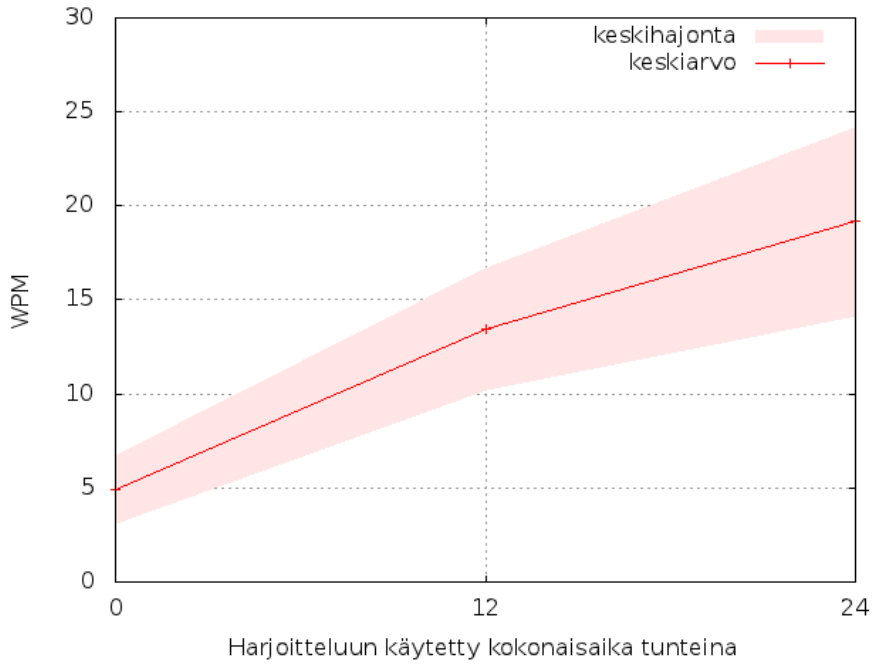
8.1.2 Koejärjestely

Ensimmäisessä käyttäjätestissä kohenkilöt kirjoittivat Stipellä yhteensä 24 tuntia useamman tunnin pituisissa istunnoissa jaettuna useille päiville. Käytännön syistä koehenkilöiden istuntojen pituudet vaihtelivat merkittävästi istunnosta ja koehenkilöstä toiseen. Harjoitusistunnot eivät myöskään aina osuneet peräkkäisille päiville. Koehenkilöille maksettiin harjoittelusta kymmenen euroa tunnilta.

Ensimmäisen istunnon aluksi koehenkilöitä ohjeistettiin suullisesti kertomalla, että käyttäjätestien tarkoitus on harjoituttaa koehenkilöitä kirjoittamaan peliohjaimella mahdollisimman nopeasti. Koehenkilöille kerrottiin, että kirjoittaminen perustuu sauvaohjaimilla suoritettaviin yksinkertaisiin eleisiin ja näytettiin muutama mallisuoritus. Koehenkilöille annettiin paperilla eleaakkostot opettelua varten.

Sekä kirjoitustehtävät että nopeusmittaukset suoritettiin kuvan 8.1 ohjelmalla. Ohjelman käyttöliittymä koostuu pääasiallisesti kahdesta tekstialueesta, joista ylempään kopioidaan ennen kirjoitustestin aloittamista teksti, joka pyritään kirjoittamalla kopioimaan alempaan tekstialueeseen. Ohjelma värjää kopioidun tekstin punaiseksi kirjoitusvirheen jälkeen. Kirjoitustesti päättyy automaattisesti, kun kopioitu teksti vastasti merkilleen lähdetekstiä. Kirjoitustestin pystyy lopettamaan myös pikanäppäimellä. Kirjoitustestin lopuksi ohjelma näyttää käyttäjälle kirjoitusnopeuden ja virhemäärät.

Kirjoitustestit olivat siis klassisia jäljennöstehtäviä. Koehenkilöitä ohjeistettiin etsimään verkosta suomen kielisiä asiatekstejä ja harjoittelemaan kopioimalla teksteistä yksittäisiä lauseita tai lyhyitä muutaman virkkeen mittaisia kappaleita. Hyviksi lähdeteksteiksi osoittautuivat uutissivustojen verkkouutiset. Koehenkilöitä kehotettiin kir-



Kuva 8.2: Ensimmäisen käyttäjätestin koehenkilöiden huippukirjoitusnopeuksien kehitys.

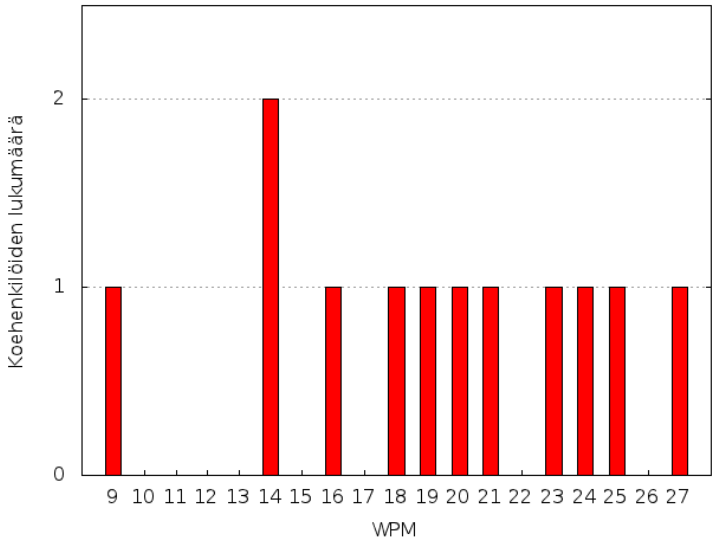
joittamaan kopioitava teksti mahdollisimman nopeasti ja mahdollisimman virheettömästi. Heitä kehotettiin korjaamaan virheet välittömästi niiden havaitsemisen jälkeen.

8.1.3 Tulokset

Koehenkilöiden huippukirjoitusnopeudet mitattiin ensimmäisen käyttäjätestin aikana kolme kertaa. Ensimmäinen mittaus suoritettiin heti ensimmäisen istunnon aluksi, vain muutaman minuutin tutustumisen jälkeen. Alkuohjeistuksen jälkeen suoritettu tutustuminen kattoi muutaman eleen suorittamisen ja elekartaston silmäilyn notaation omaksumiseksi. Toisen mittauksen huippukirjoitusnopeudeksi kirjattiin sen harjoitusistunnon nopein kirjoitusnopeus, minkä aikana 12 tunnin kokonaisharjoitteluaika tuli täyteen. Kolmannen mittauksen tuloksiksi kirjattiin viimeisten harjoitusistuntojen parhaat tulokset. Koehenkilöitä pyydettiin lisäksi ilmoittamaan heidän harjoittelun aikana havaitsemansa ohjelmavirheet ja huomiot Stipen toiminnasta.

Kuvassa 8.2 on esitettynä koehenkilöiden kirjoitusnopeuksien kehitys ensimmäisen käyttäjätestin aikana. Koehenkilöiden lähtötason kirjoitusnopeuksien keskiarvo oli 4,92 wpm (σ=1,80) ja 24 tunnin harjoittelun jälkeen 19,17 wpm (σ=5,05).

Kuvassa 8.3 on viimeisen mittauksen huippukirjoitusnopeuksien diskreetti jakauma. Se osoittaa, että käyttäjien huippukirjoitusnopeudet jakautuivat kohtuullisen tasaisesti



Kuva 8.3: Ensimmäisen käyttäjätestin koehenkilöiden huippukirjoitusnopeuksien diskreetti jakauma kolmannen mittauksen jälkeen.

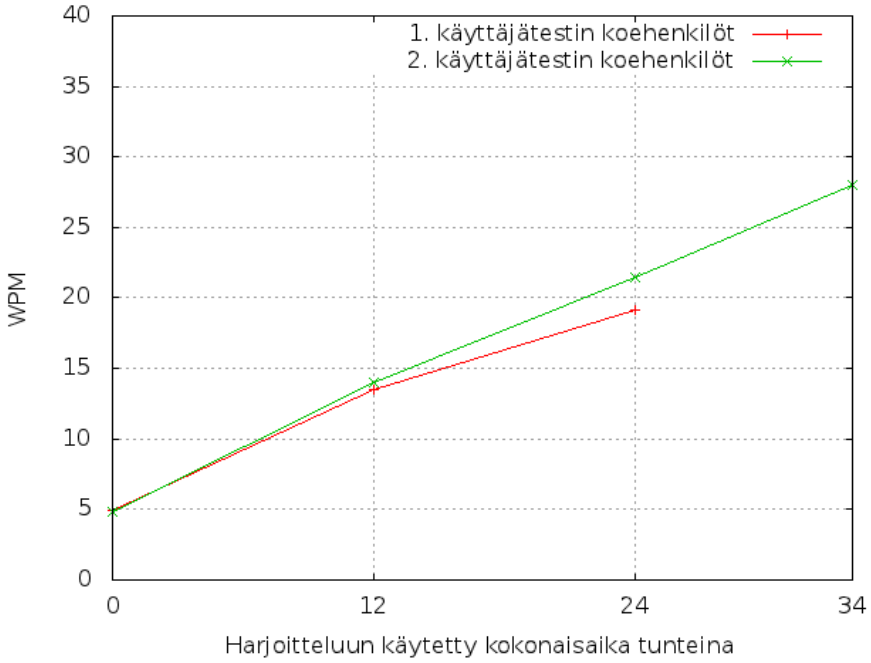
14 ja 27 wpm:n välille. Vain yhdellä käyttäjällä oli selviä vaikeuksia oppimisen kanssa. Kahden ensimmäisen mittauskerran kirjoitusnopeuksien jakaumat ovat liitteissä C.6 ja C.7.

8.2 Toinen käyttäjätesti

Toinen käyttäjätesti suoritettiin kuuden ensimmäisessä käyttäjätestissä mukana olleen koehenkilön voimin. Toisen käyttäjätestin kokonaisharjoitteluaika oli kymmenen tuntia. Koehenkilöille maksettiin sama tuntikorvaus, kymmenen euroa, kuin ensimmäisessäkin käyttäjätestissä. Toisessa käyttäjätestissä koehenkilöiden harjoittelu pyrittiin jakamaan usealle päivälle korkeintaan kahden tunnin pituisiin harjoitusistuntoihin. Istuntoja ei kuitenkaan kyetty järjestämään säännöllisesti peräkkäisille päiville käytännön syistä johtuen.

Ensimmäisessä käyttäjätestissä käytetty testiohjelma värjäsi kirjoitettavan tekstin punaiseksi mikäli kirjoitettava teksti poikkesi kopioitavasta tekstistä. Toisin sanoen, kuvassa 8.1 näkyvän ohjelma alemman tekstialueen teksti muuttui punaiseksi kirjoitusvirheen jälkeen kunnes kyseinen virhe oli korjattu. Toiseen käyttäjätestiin värjäysominaisuus päätettiin poistaa, koska sen ajateltiin korostavan liiaksi virheiden havaitsemista.

Toisen käyttäjätestin järjestelyt olivat muutoin samanlaiset ensimmäisen käyttäjätestin kanssa. Käyttäjiä pyydettiin harjoittelemaan edelleen jäljentämällä vapaavalin-



Kuva 8.4: Toiseen käyttäjätestiin osallistuneiden koehenkilöiden huippukirjoitusnopeuksien kehitys on nopeampaa.

taisia lauseita tai lyhyitä tekstikappaleita.

8.2.1 Tulokset

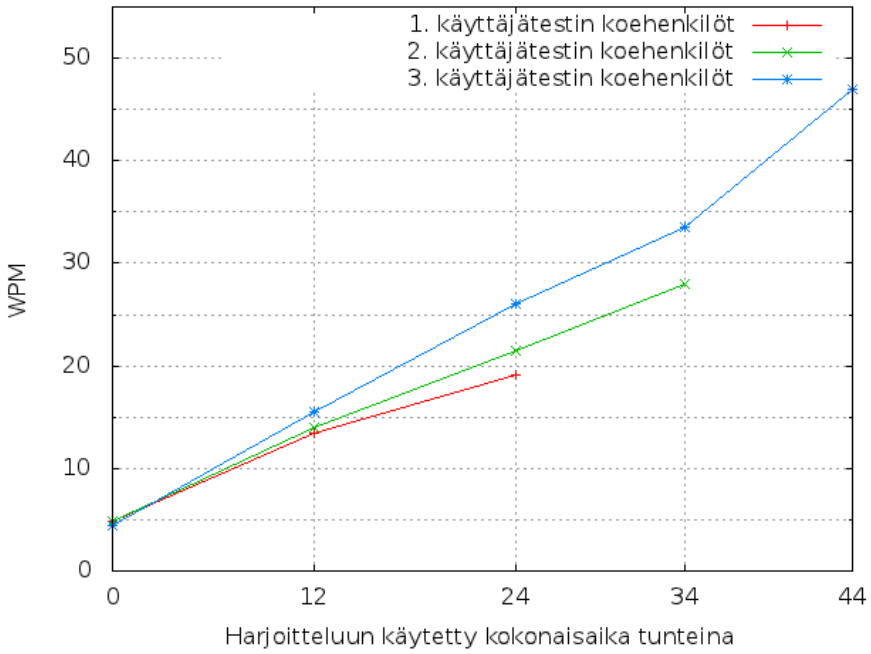
Toisen käyttäjätestin kirjoitusnopeudet kerättiin kymmenen tunnin harjoitusjakson päätteeksi. Kirjoitusnopeudeksi kirjattiin ensimmäisen käyttäjätestin tavoin viimeisen harjoitusistunnon paras tulos. Kirjoitusnopeuksien kehitys on esitettynä kuvassa 8.4. Toiseen käyttäjätestiin osallistuneiden koehenkilöiden kirjoitusnopeus 34 tunnin harjoittelun jälkeen oli 28,0 wpm (σ=5,35). Kirjoitusnopeuden kehitys jatkui toisen käyttäjätestin aikana samankaltaisena kuin ensimmäisen testin aikana.

8.3 Kolmas käyttäjätesti

Kolmanteen käyttäjätestiin valittiin kaksi toisen käyttäjätestin nopeinta kirjoittajaa. Koehenkilöitä pyydettiin kirjoittamaan kokonaisuudessaan kymmenen tuntia ja korkeintaan yksi tunti päivässä. Kolmannen käyttäjätestin harjoitusjakso kesti kokonaisuudessaan kaksi kuukautta, jonka aikana koehenkilöt pitivät lähes kuukauden pituisen tauon harjoittelusta.

Kolmas käyttäjätesti		
Suure	**Koehenkilö 8**	**Koehenkilö 12**
Kokonaisharjoitteluaika tunteina	8,03	12,15
Kirjoitustestien pituuksien ka.	26,71 (σ=11,67)	35,89 (σ=3,38)
Kirjoitustestien lukumäärien ka.	23,89 (σ=15,83)	40,15 (σ=8,83)
Harjoitusistuntojen N	18	20
Kirjoitustestien N	430	803
Vasemman käden eleiden N	34545 (41,5%)	61358 (41,6%)
Oikean käden eleiden N	48776 (58,5%)	86039 (58,4%)
Yhteensä N	83321 (100,0%)	147397 (100,0%)
Suorien eleiden N	52732 (63,29%)	93997 (63,77%)
Lyhyiden kiertoeleiden N	30025 (36,04%)	53229 (36,11%)
Pitkien kiertoeleiden N	564 (0,67%)	171 (0,12%)
Ensimmäisen harjoitusistunnon WPM-ka.	21,57 (σ=1,65)	20,72 (σ=2,66)
Viimeisen harjoitusistunnon WPM-ka.	37,04 (σ=3,64)	40,10 (σ=4,61)
Ensimmäisen harjoitusistunnon TotalER-ka.	11,77% (σ=1,74)	9,60% (σ=2,98)
Viimeisen harjoitusistunnon TotalER-ka.	4,26% (σ=1,98)	6,83% (σ=2,64)
Kaikkien harjoitusistuntojen TotalER-ka.	7,66% (σ=2,25)	6,75% (σ=1,25)

Taulukko 8.1: Kolmannen käyttäjätestin tilastojen yhteenveto.

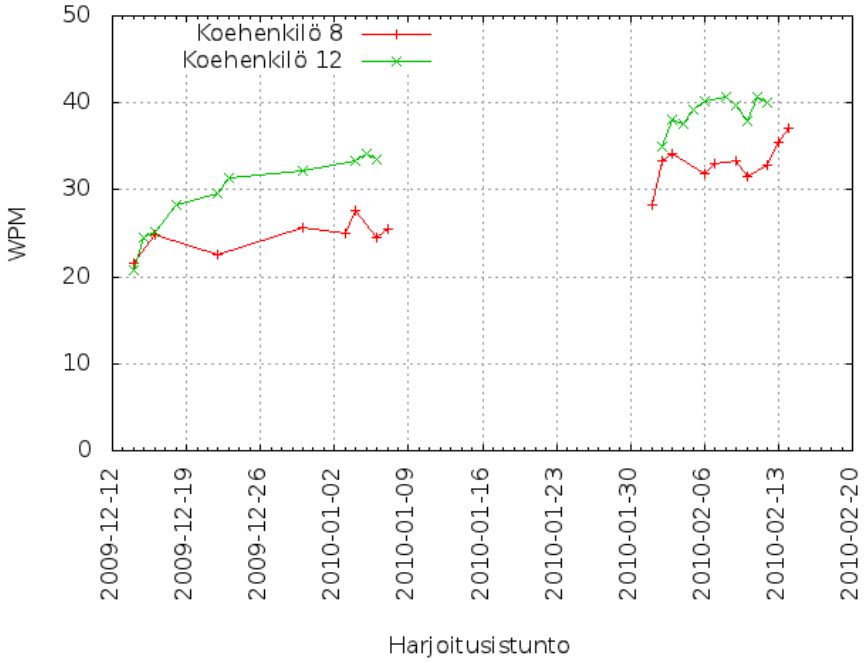


Kuva 8.5: Kolmannen käyttäjätestin koehenkilöiden huippukirjoitusnopeuksien kehitys nopeinta lähes koko harjoittelun ajan.

8.3.1 Tulokset

Kolmannen käyttäjätestin keskeiset tiedot on esitettynä taulukossa 8.1. Kohenkilö 8 jäi tavoitellusta 10 tunnin kokonaisharjoitusajasta noin kaksi tuntia. Koehenkilö 12 toisaalta ylitti tavoitellut kokonaisharjoitusajan kahdella tunnilla. Koehenkilön 12 harjoitusistuntojen pituudet olivat myös keskimäärin noin 34% pidempiä ja vaihtelivat vain vähän (σ=3,38) verrattuna koehenkilön 8 harjoitusistuntojen pituuksiin (σ=11,67). Molempien koehenkilöiden harjoitusistuntojen pituuksien sekä kokonaisharjoitteluaikojen eroihin asetetusta tavoitteesta on syynä kontrolloimaton koeympäristö. Koehenkilöt suorittivat harjoittelun kolmannessa käyttäjätestissä kotonaan ilman erillistä kokeen valvontaa. Siltikin, noin puolen tunnin pituisten harjoitusistuntojen voidaan katsoa palvelevan erittäin hyvin oppimisen seurantaa. Kolmannen käyttäjätestin harjoitusistunnot olivatkin pituuksiensa puolesta huomattavasti lähempänä aiemmissa tutkimuksissa esitettyjä harjoitusistuntojen pituuksia.

Molempien koehenkilöiden lähtötaso kolmanteen käyttäjätestiin oli hyvin samankaltainen. Molemmat kirjoittivat hieman yli 20 wpm:n nopeudella harjoitusjakson alussa, mutta koehenkilön 12 kirjoitusnopeus kehittyi käyttäjätestin aikana hieman nopeammin. Kuten kuvasta 8.5 käy ilmi, molempien koehenkilöiden huippukirjoitusnopeuksien kehitys oli nopeampaa kolmannen käyttäjätestin aikana kuin heidän aiempien

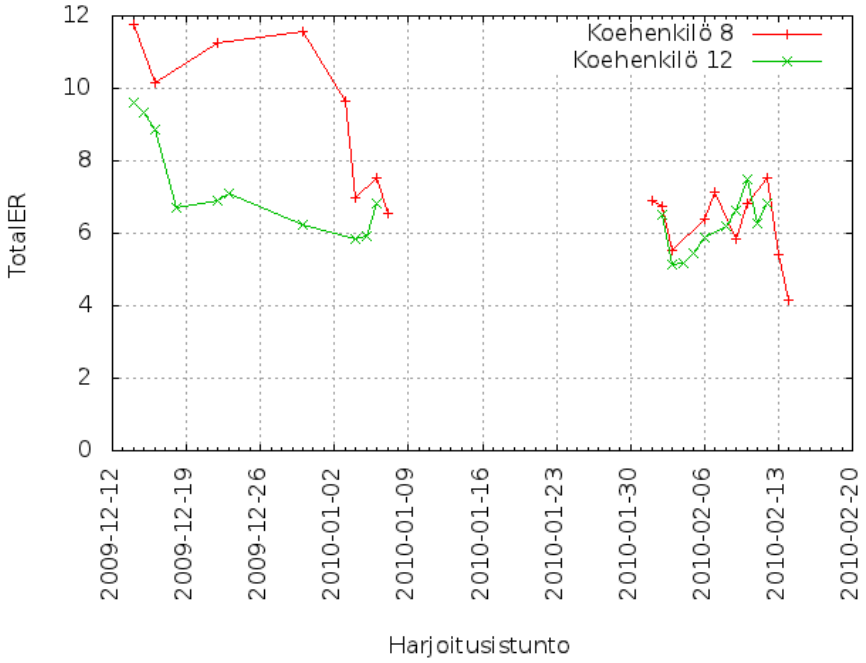


Kuva 8.6: Kolmannen käyttäjätestin koehenkilöiden harjoitusistuntojen keskimääräiset kirjoitusnopeudet

harjoitusjaksojen aikana. Yhtenä syynä nopeampaan oppimiseen voidaan pitää huomattavasti sopivamman pitusia ja paremmin jaksotettuja harjoitusistuntoja. Molempien koehenkilöiden harjoitusistuntojen pituudet olivat keskimäärin puolen tunnin ja tunnin välissä. Havainto tukee luvussa 2.3.1 esityn Baddeleyn konekirjoitustutkimuksen tuloksia. Lyhyet harjoitteluistunnot säännöllisesti kerran päivässä mahdollistavat oppimisen nopeimmin suhteessa harjoitteluun käytettyyn aikaan.

Kuvassa 8.6 on esitettynä koehenkilöiden harjoitusistuntojen keskimääräiset kirjoitusnopeudet. Kirjoitusnopeuksien kehitystä kuvaavat käyrät katkeavat välillä käyttäjätestin kuukauden tauon ajaksi. Molempien käyrien trendi on selvästi kasvava. Erittäin mielenkiintoinen piirre kirjoitusnopeuksien kehityksessä on, että käyttäjät jatkavat tauon jälkeen vähintään samalla nopeudella mihin he lopettivat ennen taukoa. Edes lähes kuukauden pituinen tauko kirjoittamisesta ei pudottanut keskimääräistä kirjoitusnopeutta. Molempien käyttäjien kirjoitusnopeudet vieläpä kasvoivat jyrkästi kahden ensimmäisen tauon jälkeisen harjoitusistunnon aikana.

Kuva 8.7 esittää koehenkilöiden kirjoitusvirheiden kehitystä kolmannen käyttäjätestin aikana. Käyrän arvot ovat harjoitusistuntojen aikana suoritettujen yksittäisten harjoitusten virhemäärien keskiarvoja. Myös virhemäärissä on nähtävissä laskua, mutta kehitys on vaihtelevampaa kuin kirjoitusnopeuden kehitys. Kokonaisuudessaan koe-

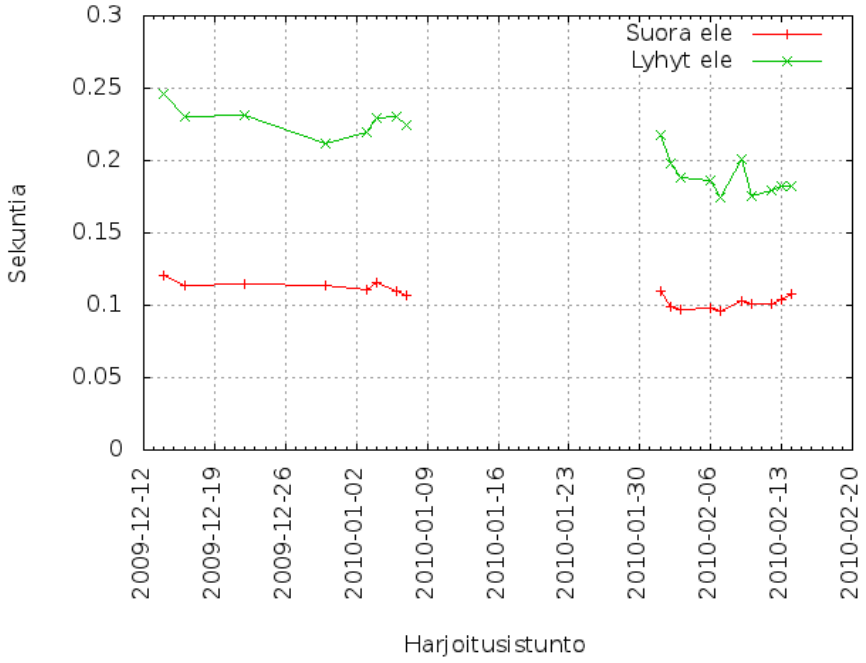


Kuva 8.7: Kolmannen käyttäjätestin koehenkilöiden harjoitusistuntojen keskimääräiset virhemäärät.

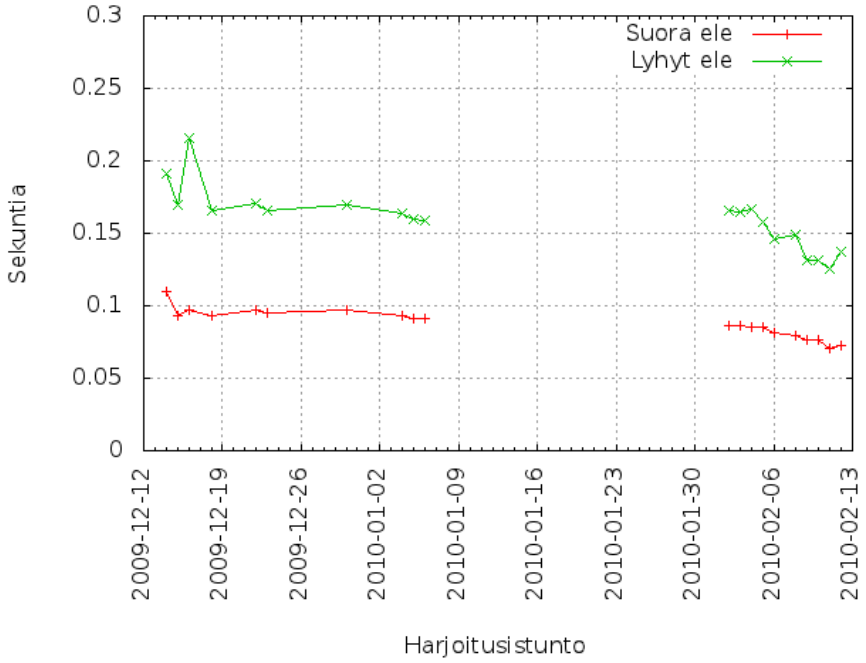
henkilöiden TotalER on 7% luokkaa, mikä on melko korkea virheprosentti verrattuna näppäimistöjen virheprosentteihin.

Kuvat 8.8 ja 8.9 esittävät koehenkilöiden eleiden suoritusaikojen kehitystä. Eleen suoritusaika on se aika, jonka sauvaohjain viettää luvussa 7.1.1 kuvatun keskusalueen ulkopuolella. Se on siis aika eleen alusta eleen loppuun. Kuvissa on esitettynä ainoastaan suorat eleet ja lyhyet kiertoeleet. Kuten taulukosta 8.1 käy ilmi, pitkien kiertoeleiden lukumäärät olivat käyttäjätestissä hyvin vähäisiä ja koehenkilöt eivät varsinaisesti harjaantuneet niiden osalta käyttäjätestin aikana. Tämän vuoksi niiden suoritusajat vaihtelivat suuresti ja eikä niitä tarkastella tässä tarkemmin. Pitkien eleiden suoritusajat ovat kuvattuna liitteissä C.1 ja C.2.

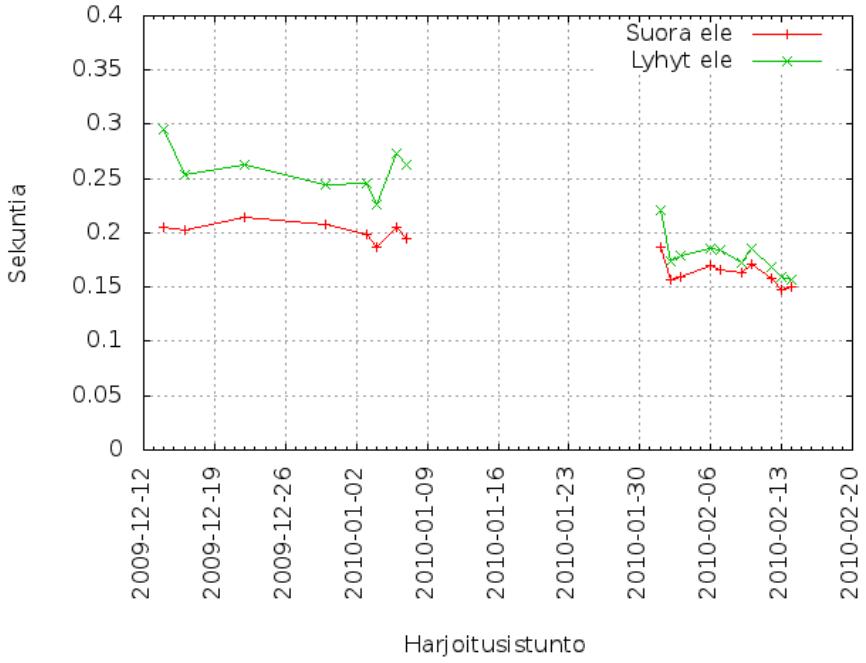
Kuvat 8.10 ja 8.11 esittävät eleiden viiveiden kehitystä. Toisin kuin eleiden suoritusaikojen kohdalla, molempien käyttäjien eleviiveiden pituudet lyhenevät ja lähestyvät harjoittelun myötä toisiaan. Suorien ja lyhyiden kiertoeleiden viiveet eroavat harjoitusjakson alussa selvästi toisistaan. Erot eleiden viiveissä olivat noin 100 ms:n luokkaa. Koehenkilön 12 eleviiveet olivat harjoitusjakson alussa hieman korkeampia kuin koehenkilön 8 eleviiveet, mutta ne putosivat nopeasti samalle tasolle. Tasaisemmin jaettu ja suurempi harjoitusmäärä selittänevät osaltaan koehenkilön 12 nopeamman kehityksen myös eleviividen osalta.



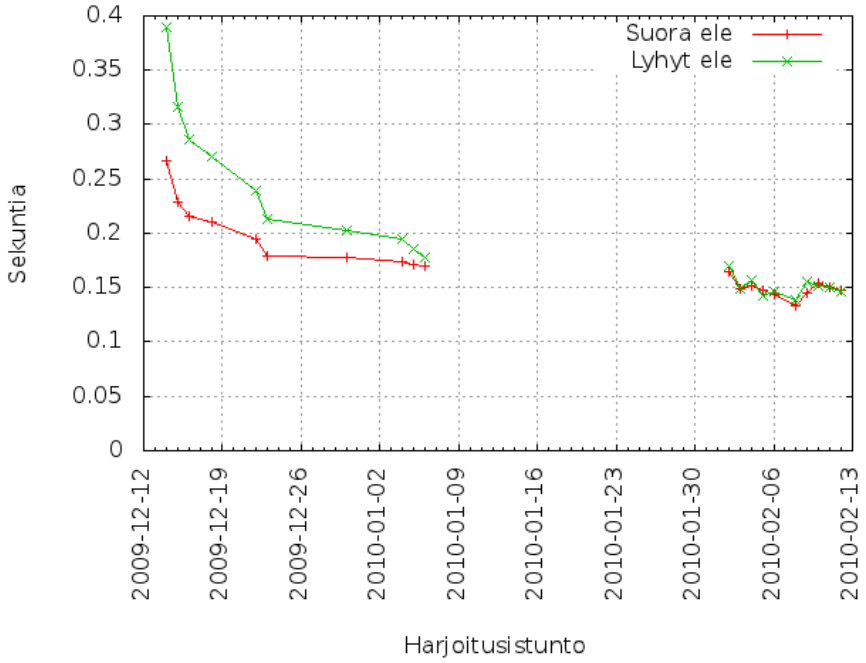
Kuva 8.8: Koehenkilön 8 kolmannen käyttäjätestin eleiden keskimääräiset suoritusajat harjoitusistunnoittain.



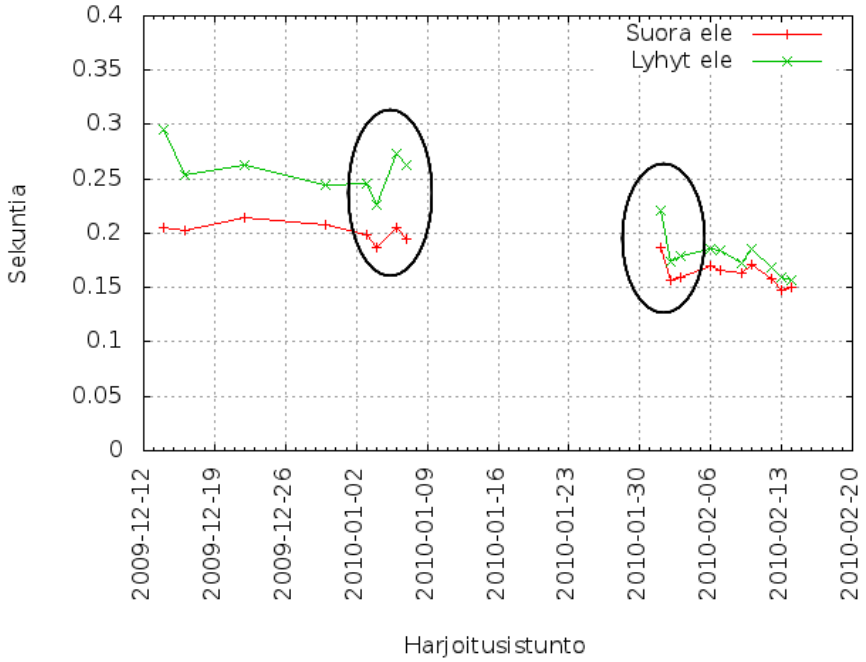
Kuva 8.9: Koehenkilön 12 kolmannen käyttäjätestin eleiden keskimääräiset suoritusajat harjoitusistunnoittain.



Kuva 8.10: Koehenkilön 8 kolmannen käyttäjätestin keskimääräiset eleviiveet harjoitusistunnoittain.



Kuva 8.11: Koehenkilön 12 kolmannen käyttäjätestin keskimääräiset eleviiveet harjoitusistunnoittain.



Kuva 8.12: Koehenkilön 8 kolmannen käyttäjätestin eleiden viiveet noudattelevat samaa kaavaa eleluokasta riippumatta.

Harjoitusjakson lopulla koehenkilön 12 suorien eleiden ja lyhyiden kiertoeleiden välillä ei ole viiveissä eroja juuri lainkaan. Koehenkilön 8 viiveidet erot olivat myös hyvin lähellä toisiaan. Molempien koehenkilöiden eleviiveet olivat harjoitusjakson lopulla 150 ms:n luokkaa. Eleiden viiveet kuvastavat sitä, kuinka paljon käyttäjä joutuu käyttämään kognitiivista prosessointia eleen suoritukseen valmistautuessaan. Usein toistetuille eleille on muodostunut vahva motorinen kartta ja eleen hakeminen muistista on nopeaa. Harvinaista merkkiä vastaavan eleen palauttaminen muistista kestää huomattavasti pidempään. Eleiden suoritusta edeltävien viiveiden lyheneminen kertoo siis oppimisesta.

Se, että suorien eleiden ja lyhyiden kiertoeleiden viiveet olivat lähes identtiset harjoitusjakson lopuksi, voidaan tulkita kertovan siitä, että koehenkilöt olivat sisäistäneet molempien eleluokkien kirjaimet. Toisin sanoen, koehenkilöille ei ollut enää välilä oliko kirjain suoran eleen vai lyhyen kiertoeleen takana. Molempien eleiden motoriset komennot oli yhtä nopeaa hakea muistista.

Molempien koehenkilöiden eleviiveiden kehityksessä on havaittavissa vielä eräs mielenkiintoinen ilmiö. Sekä suorien eleiden että lyhyiden kiertoeleiden viiveet vaihtelevat samankaltaisesti harjoitusistunnoittain. Käyrien perusteella näyttäisi, että eleen suorittamiseen valmistautuminen on osin riippumatonta suoritettavasta eleestä. Esimerkiksi

kuvassa 8.12 koehenkilön molempien eleluokkien viiveet noudattelevat samaa kaavaa ympyröidyillä alueilla.

9 Pohdinta

Pääasiallinen tavoite tutkimuksessa oli kehittää perinteisille peliohjaimille sauvaohjaimien eleisiin perustuva tekstinsyöttömenetelmä, joka mahdollistaa suuren kirjoitusnopeuspotentiaalin. Luvussa 7 esitelty Stipe on aiempien eleisiin perustuvien tekstinsyöttömenelmien toimintaperiaatteita uudella tavalla yhdistelevä tekstinsyöttömenetelmä perinteisille peliohjaimille.

Käyttäjätestien tulokset osoittivat, että aivan ensimmäisten kokeilukertojen kirjoitusnopeus Stipe-menetelmällä on kohtuullisen vaatimaton muiden eleisiin perustuvien menetelmien tavoin. Kaikkien koehenkilöiden kirjoitusnopeudet alkumittauksessa olivat alle 10 wpm, keskimäärin hieman alle 4,92 wpm. TwoStickin alkuhetkien kirjoitusnopeus on samaa luokkaa, 5,1 wpm, ja EdgeWriten 6,40 wpm [27, 58]. Virtuaalisten valintanäppäimistöjen kirjoitusnopeudet ovat luokkaa 6-7 wpm.

Stipen oppimiskäyrän vertailu aiempiin menetelmiin on vaikeaa täysin erilaisen harjoittelujaksotuksen vuoksi. Esimerkiksi Költringerin ja Isokosken TwoStick-menetelmän käyttäjätesteissä koehenkilöt kirjoittivat 20× 15min harjoitusistuntoa, yhteensä siis viisi tuntia [27]. Stipen ensimäisessä käyttäjätestissä koehenkilö saattoi kirjoittaa saman viisi tuntia yhdessä istunnossa. On selvää, että lyhyempiin istuntoihin jaksotettu harjoittelu nopeuttaa oppimista huomattavasti. TwoStick-käyttäjien kirjoitusnopeudet harjoitusjakson jälkeen olivat keskimäärin 14,9 wpm. Stipe-käyttäjät saavuttivat saman nopeuden ensimmäisen käyttäjätestin aikana noin 12 tunnin harjoittelun jälkeen.

Aiempien tekstinsyöttömenetelmien käyttäjätesteisssä ei ole suoritettu pitkän aikavälin seurantaaa. Käyttäjätestit käsittävät yleensä alle 10 tuntia harjoittelua koehenkilöä kohden. Stipen käyttäjätesteissä kahden kohenkilöiden kehittymistä seurattiin noin 44 tuntia. Kalenteriajassa 44 tunnin harjoittelu jakaantui usean kuukauden ajalle. Tässä tutkimuksessa oli tavoitteena selvittää Stipen kirjoitusnopeuspotentiaali. Pitkäaikainen seuranta mahdollisti potentiaalin luotettavaan selvittämisen.

Tulokset osoittivat, että Stipellä on selvästi mahdollista päästä aiempiin tekstinsyöttömenetelmiin nähden suuriin kirjoitusnopeuksiin. Koehenkilöt pystyivät kirjoittamaan viimeisen käyttäjätestin lopulla jopa yli 40 wpm:n nopeudella. Aiempien peliohjainmenetelmien tutkimuksissa on mitattu pääsääntöisesti alle 20 wpm:n kirjoitusnopeuksia.

Kirjoitusvirheiden määrä (TotalER) kolmannen käyttäjätestin koehenkilöillä oli suurehkosta kirjoitusnopeudesta huolimatta melko korkea, ollen noin 7% luokkaa. TwoS-

tickin kirjoitusvirheiden määrä 20×15min harjoittelun jälkeen oli 8,2% [27]. EdgeWriten kirjoitusvirheet olivat aivan aloittelijoilla 10,85% [58]. Virtuaalisten valintanäppäimistöjen kirjoitusvirheet ovat tyypillisesti luokkaa 3-5%. On huomattava, että Stipen 7% TotalER-taso oli saavutettu pitkän harjoitteluajan jälkeen. Aloittelijoiden kohdalla virheprosentti on todennäköisesti tätä korkeampi. On hyvin mahdollista, että Stipen alkuvaiheen kirjoitusvirheiden määrä on esimerkiksi TwoStickiä ja EdgeWriteä korkeampi.

Tulokset osoittavat, että Stipe omaa kohtuullisen suuren kirjoitusnopeuspotentiaalin, mutta menetelmän opettelu on kohtuullisen hidasta. Mikäli suuren kirjoitusnopeuden saavuttamiseen vaaditaan suuri työmäärä, käyttäjät eivät helposti innostu uuden menetelmän opettelusta. Jatkokehityksen kannalta onkin keskeistä nopeuttaa oppimisprosessia ja tehdä siitä houkuttelevampaa. Oppiminen voi nopeutua tunnistusalgoritmin tarkkuutta parantamalla, mutta myös muita tapoja on syytä etsiä. Erityisesti alkuvaiheen oppimista pitäisi tehostaa.

9.1 Käyttöliittymä

Stipen kehityksen peruslähtökohta oli määrittää riittävän suuri eleiden joukko tavanomaisimpien kirjaimien ja välimerkkien kattamiseksi yhdellä eleaakkostolla. Tämän voidaan katsoa olevan tyypillinen tapa ei-ikonisiin eleisiin perustuvien tekstinsyöttömenetelmien kohdalla. Ensin laaditaan riittävän monta toisistaan hyvin eroavaa elettä, jonka jälkeen halutut kirjaimet liitetään eleisiin muodostaen eleaakkoston. Kirjaimien sovittamisessa eleaakkostoon on luonnollista käyttää kirjainten esiintymistiheyksiä. Yleisimmät kirjaimet tuotetaan helpoimmilla eleillä ja harvinaisimmat vaikeimmilla eleillä. Näin tehtiin myös Stipen kohdalla. Lopputuloksena syntyi kuitenkin eleaakkosto, joka ei ollut käyttäjille entuudestaan tuttu eikä kovin intuitiivinen. Intuitiivisuuden puute hidastaa aloittelijoiden oppimista.

Alkuvaiheen opettelua ja intuitiivisuutta voitaisiin lisätä tuomalla toteutuksesta tuttu näppäimistöanalogia myös käyttöliittymään. Eleaakkosto näytettäisiin käyttäjälle vain perinteisenä QWERTY-näppäimistönä. Hahmotelma tällaisesta näppäimistöstä on esitettynä kuvassa 9.1. Näppäimistössä on tavallisimmat latinalaisen aakkoston kirjaimet, muutama välimerkki ja numerot. Näppäimistö on itseasiassa hyvin tavanomainen virtuaalinen valintanäppäimistö. Kuvan sininen neliö on kohdistin, jolla käyttä voi navigoida näppäimistöllä tavallisen virtuaalisen näppäimistön tapaan. Tämän lisäksi näppäimistö on jaettu puoliksi molemmille käsille ja jokaiseen näppäimeen on piirretty Stipen ele. Näppäimistö toimii siten samanaikaisesti sekä perinteisenä virtuaalisena näppäimistönä että Stipen eleaakkostona.

Kuva 9.1: Hahmotelma halkaistusta Stipen QWERTY-näppäimistöstä.

Perinteiseen virtuaalisen näppäimistöön pohjaava käyttöliittymä saattaisi pehmentää laskua uuteen tekstinsyöttömenetelmään. Käyttäjä voisi edelleen kirjoittaa navigoimalla osoitinta näppäimistön päällä valiten kirjaimia yksitellen nuolinäppäimien avulla, mutta hänen olisi myös mahdollista käyttää "oikopolkua" suorittamalla näppäintä vastaava ele sauvaohjaimilla. Tällaisen lähestymistavan voisi arvella madaltavan peliohjaimien eleisiin perustuvien tekstinsyöttömenetelmien käyttöönottoa merkittävästi.

9.2 Käyttäjätestit

On selvää, että kaksi ensimmäistä käyttäjätestiä eivät olleet oppimisnopeuden kannalta tarkasteltuna optimaalisesti jaksotettuja. Harjoitusistuntojen pituudet olivat myös merkittävästi pidempiä kuin aiempien tutkimusten käyttäjätesteissä. Tästä syystä tuloksia oli vaikea verrata aiempien tutkimustulosten kanssa.

Jatkossa käyttäjätestejä tulisi suunnitella lyhyemmiksi ja yksinkertaisemmiksi. Eleaakkostoista voisi ainakin alkuvaiheessa jättää pitkät kiertoeleet kokonaan pois. Suorilla eleillä ja lyhyillä kiertoeleillä pystyy kattamaan kirjoittamisen opetteluun vaadittavat merkit. Kirjoitettava tekstin tulisi olla ainakin osan aikaa käyttäjätestistä vakio kaikkien käyttäjien kesken. Käyttäjätesteissä voitaiisiin kirjoituttaa myös vain hyvin lyhyitä lauseita tai sanoja. Tällöin koehenkilöt voitaisiin asettaa muistamaan kirjoitettava teksti ulkoa ennen testin alkua ja testin aikainen lukemisen vaikutus voitaisiin eliminoida täysin. Käyttäjätestit tulisi myös jaksottaa lyhyempiin harjoitusistuntoihin. Harjoitusistuntojen tulisi pysytellä korkeintaan yhden tunnin pituisina. Edellä mainitut toimenpiteet mahdollistaisivat myös kontrolloidumman koeasetelman ja tulosten vertailu aiempien tutkimusten tuloksiin helpottuisi.

Kuva 9.2: Peukalot ovat vinossa normaalissa peliohjainotteessa.

9.3 Tunnistusalgoritmi

Kirjoitusvirheiden määrää saattaisi olla mahdollista vähentää parantamalla tunnistusalgoritmia. Nykyinen algoritmi on toteutukseltaan kohtuullisen naiivi, vain polkujen euklidisia etäisyyksiä laskeva funktio, eikä sen tunnistustarkkuudesta ole mitattua tietoa. Algoritmin voidaan selvästi katsoa toimineen konseptin testaustarpeisiin nähden riittävän hyvin, koska käyttäjät saavuttivat testien aikana merkittävät kirjoitusnopeudet. Algoritmin mahdollinen epätarkkuus ei tehnyt kirjoittamisesta liian vaikeaa. Menetelmän kehittäminen paremmaksi vaatii kuitenkin algoritmin systemaattista tarkkuuden mittaamista ja parantamista.

Algoritmia voitaisiin kehittää esimerkiksi laskemalla ominaisuusvektoriin lisäksi muitakin elettä kuvaavia ominaisuuksia. Esimerkkeinä näistä mainittakoon eleen suorittamiseen kulunut aika, eleen kokonaispituus, eleen kaarevuus ja eleen kulmikkuus. Tällöin algoritmia voitaisiin myös muuttaa luvussa 4.3.1 kuvatun kaltaiseksi, todennäköisyyksiä laskevaksi algoritmiksi, jossa jokainen ominaisuusvektorin ominaisuus painotettaisiin, normalisoitaisiin ja laskettaisiin yhteen.

Algoritmin kehityksessä olisi myös syytä huomioida peliohjaimien ergonomia. Kuvassa 9.2 on luonnollinen ote peliohjaimesta. Kuvasta käy ilmi, että peukalot eivät ole aivan pääilmansuuntien suuntaisesti vaan joitakin asteita vinossa “sisäänpäin”. On syytä olettaa, että peukalot liikkuvat pitkin vinoja akseleita myös eleitä suoritettaessa. Peukaloiden vinot liikeradat vaikuttavat eleiden tunnistamiseen, mikä olisi syytä ottaa huomioon jatkossa tunnistusalgoritmia kehitettäessä.

9.4 Ohjelmistotototeutus

Toteutuksen ohjelmistoarkkitehtuurin puolesta näppäimistöajuria muistuttavana tekstinsyöttömenetelmä on sovitettavissa useisiin peliohjainmalleihin. Vaikka käyttäjätesteissä Stipeä koekäytettiin vain Wii Classic -peliohjaimilla, se toteutettiin myös Playstation Dualshock 3 -ohjaimille. Luvussa 7 esitetyn ohjelmistoarkkitehtuurin mukaisesti, ainoa ero Wii Classic -toteutukseen on eritavoin toteutettu *Stipepad*-luokan aliluokka, *DualShock3*. *DualShock3*-luokka ottaa huomioon luvussa 5 esitetyt Sonyn peliohjaimen tekniset yksityiskohdat ja eroavaisuudet Wii Classic -ohjaimeen nähden.

Nykyinen arkkitehtuuri kuvaa Stipen yksinkertaisena funktiona, joka saa syötteenä peliohjaintapahtumia, jotka Stipe kuvaa näppäimistötapahtumiksi. Stipen arkkitehtuuria voisi kehittää lisäämällä Stipe-komponenttiin niinsanotun sovellus-portin, jota kautta Stipe-tietoiset -sovellukset voisivat saada käyttöönsä pelkkien näppäintapahtumien lisäksi tunnistusalgoritmin tuloksia ja tulkita niitä eri tavoin sovelluksen kontekstista riippuen. Esimerkiksi tekstinsyöttöohjelmat voisivat käyttää sanakirjaan perustuvaa menetelmää virheiden korjaamiseksi. Sovellukset voisivat valita esimerkiksi algoritmin palauttamista tuloksista toiseksi todennäköisimmän eleen, mikäli sanakirjakontekstista saatava vihje kertoisi sen olevan merkittävästi todennäköisempi kuin Stipen algoritmin ensisijainen valinta. Vastaavasti numerosyötteitä odotettaessa Stipe-tietoinen sovellus voisi tulkita virheellisiä eleitä todennäköisemmin numeroiksi.

Suorituskyvyn osalta nykyisen toteutuksen voidaan katsoa olevan riittävän tehokas. Stipe toimii ilman mitään suorituskykyongelmaa jopa verrattain heikkotehoisessa Asus Eee PC 900 -minikannettavassa [1].

[1] http://event.asus.com/eeepc/comparison/eeepc_comparison.htm

10 Yhteenveto

Tekstinsyöttö on yksi keskeisimmistä ihmisen ja tietokoneen välisistä vuorovaikutusmenetelmistä. Suuri osa digitaalisesta informaatiosta on luonteeltaan tekstuaalista. Työpöytäkoneiden tekstinsyöttömenetelmänä perinteinen näppäimistö on laajimmin levinnyt ja osoittanut olevansa sekä tehokas niin ammattimaisessa käytössä että intuitiivinen aloittelevien käsissä.

Mobiililaitteiden yleistyessä on tarvittuu uusia vaihtoehtoisia tapoja syöttää tekstiä. Pienikokoiset näppäimistöt ovat olleet pitkään luonnollinen ratkaisu kämmentietokoneissa ja matkapuhelimissa. Puheentunnistusta on odotettu auttamaan tekstinsyötön haasteissa mobiiliympäristöissä. Uudet muodikkaat monikosketusnäytöt ovat mahdollistaneet lukuisia erilaisia virtuaalinäppäimistöjä ja eletunnistusmenetelmiä. Mobiililaitteiden tekstinsyöttömenetelmistä käydäänkin kovaa kilpailua.

Pelikonsolit ja erilaiset peliohjaimet ovat kuitenkin jääneet hieman taka-alalle tekstinsyöttömenetelmien kehityksessä. Nykyisin vallitsevat pelikonsolien tekstinsyöttömenetelmät ovat kömpelöt ja hitaat virtuaaliset valintanäppäimistöt. On selvää, että ne eivät pysty tarjoamaan ratkaisua moderneille tekstinsyötön tarpeille. Sosiaalisen median palvelut sekä pelien pikaviestintäominaisuudet vaativat nopeaa ja vaivatonta tekstinsyöttöä. Sekä useat laitevalmistajat että sisällöntuottajat ovat huomanneet ja ratkaisseet tekstinsyöttöongelman minimoimalla tekstuaalisen sisällön pelikonsolituotteissaan. Helposti opittavan nopean ja vaivattoman tekstinsyöttömenetelmän löytyminen pelikonsoleihin tarjoaisi uutta kaupallista arvoa sekä uusia ulottuvuuksia olohuoneiden viihdekeskuksiin.

Tässä tutkielmassa esitetyn tutkimuksen tavoitteena oli kehittää perinteisille peliohjaimille tekstinsyöttömenetelmä, jonka potentiaalinen kirjoitusnopeus olisi vallitsevia peliohjainten tekstinsyöttömenetelmiä selvästi suurempi. Tutkielmassa kuvattiin uuden sauvaohjaimilla suoritettaviin eleisiin perustuvan tekstinsyöttömenetelmän, Stipen, toimintaperiaatteet sekä ohjelmistoarkkitehtuuri. Tutkielmassa esiteltiin lisäksi tutkimuksen puitteissa suoritettujen käyttäjätestien tulokset ja ajatuksia menetelmän jatkokehityksestä.

Käyttäjätestien tulokset osoittavat, että kehitetty menetelmä omaa suuren kirjoitusnopeuspotentiaalin. Menetelmää pitkään käyttäneet koehenkilöt pääsivät yli 40 wpm:n kirjoitusnopeuksiin, mikä on huomattavasti enemmän mitä aiemmat perinteisille peliohjaimille suunnitellut tekstinsyöttömenetelmät antavat odottaa. Koehenkilöi-

den kirjoitusvirheiden määrä jäi kuitenkin hieman korkealle, noin 7% (TotalER) tasolle. Muihin eleisiin perustuviin menetelmiin nähden luku ei ole korkea, mutta on selvästi yksi osa-alue, jossa menetelmää olisi mahdollista kehittää. Yksi jatkotutkimuksen kohteista tulisikin olla tunnistusalgoritmin systemaattinen tunnistustarkkuuden parantaminen.

Menetelmän erilaisuus ja opetteluun vaadittava työmäärä monien muiden eleisiin perustuvien tekstinsyöttömenetelmien tavoin muodostaa kuitenkin suurimman haasteen Stipen käyttöönotolle. Yli 40 wpm:n nopeudella kirjoittaneet koehenkilöt harjoittelivat yhteensä noin 44 tuntia. Vaikka menetelmän eleiden liikkeet sinänsä ovat yksinkertaisia suorittaa ja omaksua, voi menetelmän erilaisuus intuitiivisiin virtuaalisiin näppäimistöihin nähden vaikeuttaa menetelmän opettelua alkuvaiheessa. Eräs houkutteleva oppimista tehostava suuntaus voisi olla tutun QWERTY-analogian korostaminen käyttöliittymällä, joka esittäisi eleet näppäinvalintoina. Tutkimuksessa löydetyn rohkaisevan kirjoitusnopeuspotentiaalin saavuttamiseksi nopeammin tarvitaankin lisää tutkimusta erityisesti menetelmän alkuvaiheen oppimisen arvioinnissa ja käytettävyyden kehittämisessä.

11 Lähteet

[1] Cwiid: A collection of linux tools written in c for interfacing to the nintendo wiimote. http://cwiid.org/. 51

[2] Device class definition for hid 1.11 - universal serial bus (usb). http://www.usb.org/developers/devclass_docs/HID1_11.pdf. 53

[3] Extending and embedding the python interpreter. http://docs.python.org/release/2.6.6/extending/index.html. 66

[4] Libcwiid: A library written in c for interfacing to the nintendo wiimote. http://abstrakraft.org/cwiid/wiki/libcwiid. 51

[5] Nintendo: Company history. http://www.nintendo.com/corp/history.jsp. 48

[6] Python programming language - official website. http://python.org/. 66

[7] Wiibrew: Wiimote. http://wiibrew.org/wiki/Wiimote/. 49, 51

[8] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. Analysis of text entry performance metrics. In *In Proc. IEEE TIC-STH 2009. IEEE*, pages 100–105, 2009. 17, 18, 19, 21

[9] Tom Bellman and Scott I. Mackenzie. A Probabilistic Character Layout Strategy for Mobile Text Entry. In *Graphics Interface*, pages 168–176, 1998. 56

[10] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25 –30, January 1965. 74

[11] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983. 26

[12] Steven J. Castellucci and I. Scott MacKenzie. Graffiti vs. unistrokes: an empirical comparison. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 305–308, New York, NY, USA, 2008. ACM. 2, 27, 28, 29, 30, 63

[13] Steven J. Castellucci and I. Scott MacKenzie. Unigest: text entry using three degrees of motion. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 3549–3554, New York, NY, USA, 2008. ACM. 3, 58

[14] Stanley F. Chen. *Building probabilistic models for natural language*. PhD thesis, Harvard University, Cambridge, MA, USA, 1996. UMI Order No. GAX96-31460. 15

[15] Edward Clarkson, James Clawson, Kent Lyons, and Thad Starner. An empirical study of typing rates on mini-qwerty keyboards. In *In CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1288–1291. ACM Press, 2005. 24

[16] Baddeley A. D. *Human Memory: Theory and Practice*. Psychology Press, 1997. 13

[17] Baddeley A. D. and Longman D. J. A. The influence of length and frequency of training session on the rate of learning to type. *Ergonomics*, 21:627–635, 1978. 2, 12, 13

[18] Douglas C. Engelbart. X-y position indicator for a display system. United States Patent Office, November 1970. Patent number 3541541. 45

[19] David Goldberg and Cate Richardson. Touch-typing with a stylus. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 80–87, New York, NY, USA, 1993. ACM. 27, 63

[20] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. Language modeling for soft keyboards. In *Proc. AAAI 2002*, pages 419–424. AAAI Press, 2002. 16

[21] G. Hohpe and B. Woolf. *Enterprise integration patterns: designing, building, and deploying*. Addison-Wesley, 2004. 68

[22] P. Isokoski. *Manual Text Input: Experiments, Models, and Systems*. PhD thesis, University of Tampere, 2004. 15, 16, 17, 18, 22, 23, 24

[23] Poika Isokoski and Roope Raisamo. Quikwriting as a multi-device text entry method. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 105–108, New York, NY, USA, 2004. ACM. 71

[24] Sweatt D. J. *Mechanisms Of Memory*. Academic Press, 2010. 12

[25] Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 568–575, New York, NY, USA, 1999. ACM. 25

[26] Avi Karni, Gundela Meyer, Christine Rey-Hipolito, Peter Jezzard, Michelle M. Adams, Rober Turner, and Leslie G. Ungerleider. The acquisition of skilled motor performance: Fast and slow experience-driven changes in primary motor cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 95(3):861–868, February 1998. 13

[27] Thomas Költringer, Poika Isokoski, and Thomas Grechenig. Twostick: writing with a game controller. In *GI '07: Proceedings of Graphics Interface 2007*, pages 103–110, New York, NY, USA, 2007. ACM. 3, 55, 56, 57, 58, 95, 96

[28] Thomas Költringer and Thomas Grechenig. Comparing the immediate usability of graffiti 2 and virtual keyboard. In *Extended Abstracts of ACM CHI*, pages 1175–1178. ACM Press, 2004. 30

[29] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965. Original in Russian – translation in Soviet Physics Doklady 10(8):707-710, 1966. 20

[30] I. Scott MacKenzie. Kspc (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, Mobile HCI '02, pages 195–210, London, UK, 2002. Springer-Verlag. 19, 23

[31] I. Scott MacKenzie. Evaluation of text entry techniques. In *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufman, 2007. 16, 17

[32] I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. Letterwise: prefix-based disambiguation for mobile text input. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 111–120, New York, NY, USA, 2001. ACM. 23

[33] I. Scott MacKenzie and R. William Soukoreff. A model for two-thumb text entry. In *Proceedings of Graphics Interface*, pages 117–124, 2002. 24

[34] I. Scott MacKenzie and Shawn X. Zhang. The immediate usability of graffiti. In *Proceedings of the conference on Graphics interface '97*, pages 129–137, Toronto, Ont., Canada, Canada, 1997. Canadian Information Processing Society. 28

[35] I. Scott MacKenzie and Shawn X. Zhang. The design and evaluation of a high-performance soft keyboard. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 25–31, New York, NY, USA, 1999. ACM. 16, 56

[36] Maryam Mirian, Majid Nili Ahmadabadi, Babak Araabi, and Ronald Siegwart. Comparing learning attention control in perceptual and decision space. In Lucas Paletta and John Tsotsos, editors, *Attention in Cognitive Systems*, volume 5395 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin / Heidelberg, 2009. 11

[37] Chomsky N. *Aspects of the Theory of Syntax*. MIT Press, 1965. 10

[38] Jean-Luc Nespoulous, Paul Perron, and Andre Roch Lecours. *The Biological Foundations of Gestures: Motor and Semiotic Aspects*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986. 34

[39] Walker M. P., Brakefield T., Morgan A., Hobson J. A., and Stickgold R. Practice with sleep makes perfect: Sleep-dependent motor skill learning. *Neuron*, 35(1):205–211, 2002. 13

[40] Ken Perlin. Quikwriting: continuous stylus-based text entry. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 215–216, New York, NY, USA, 1998. ACM. 30

[41] M. Pääkkönen. A:sta Ö:hön. suomen kielen yleisyystilastoja. *Kielikello*, (1), 1991. 65

[42] Kathryn M. Refshauge, S. L. Kilbreath, and S. C. Gandevia. Movement detection at the distal joint of the human thumb and fingers. *Experimental Brain Research*, 122:85–92, 1998. 10.1007/s002210050494. 48

[43] Dean Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25:329–337, July 1991. 2, 39, 40

[44] Ferreira V. S. Language production. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2010. 10, 11

[45] Katsuyuki Sakai, Okihide Hikosaka, and Kae Nakamura. Emergence of rhythm during motor learning. *Trends in Cognitive Sciences*, 8(12):547 – 553, 2004. 14

[46] Miika Silfverberg. Historical overview of consumer text entry technologies. In *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufman, 2007. 22, 23, 24, 25, 26

[47] Jyoti Snehi. *Computer Peripherals and Interfaces*. Firewall Media, 2006. 43

[48] R. William Soukoreff and I. Scott MacKenzie. Measuring errors in text entry tasks: an application of the levenshtein string distance statistic. In *CHI '01 extended abstracts on Human factors in computing systems*, CHI '01, pages 319–320, New York, NY, USA, 2001. ACM. 20

[49] R. William Soukoreff and I. Scott MacKenzie. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, pages 113–120, New York, NY, USA, 2003. ACM. 2, 17, 18, 19, 21

[50] Zeller T. A great idea that's all in the wrist. *The New York Times*, 2005. 43

[51] Paul Tibbetts. The concept of voluntary motor control in the recent neuroscientific literature. *Synthese*, 141:247–276, 2004. 10.1023/B:SYNT.0000043021.33695.99. 11

[52] Shari Trevin and John Arnott. Text entry when movement is impaired. In *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufman, 2007. 25

[53] Halsband U. and Lange R. K. Motor learning in man: A review of functionla and clininal studies. *Journal of Physiology*, 99:414–424, 2006. 12

[54] Carroll D. W. *Psychology of Language*. Brooks/Cole Publishing Company, 1994. 10

[55] Don Willems and Ralph Niels. Definitions for features used in online pen gesture recognition. Technical report, NICI, Radboud University Nijmegen, 2008. 34

[56] Andrew D. Wilson and Maneesh Agrawala. Text entry using a dual joystick game controller. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 475–478, New York, NY, USA, 2006. ACM. 3, 55, 56

[57] Jacob Wobbrock, Htet H. Aung, Brad Myers, and Edmund Lopresti. Integrated text entry from power wheelchairs. *Behaviour and Information Technology*, 24(3):187–203, June 2005. 19

[58] Jacob O. Wobbrock, Brad A. Myers, and Htet Htet Aung. Writing with a joystick: a comparison of date stamp, selection keyboard, and edgewrite. In *GI '04: Proceedings of Graphics Interface 2004*, pages 1–8, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society. 55, 56, 57, 95, 96

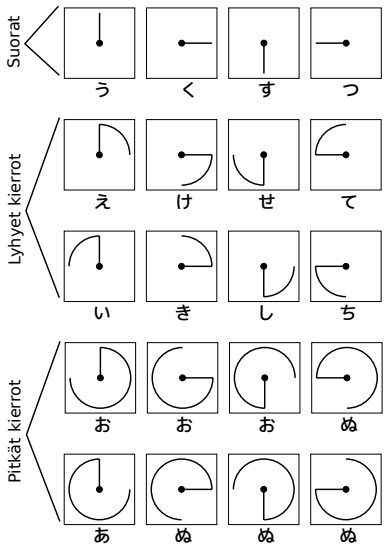
[59] Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. Edgewrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA, 2003. ACM. 19, 32, 63

[60] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM. 2, 40, 41, 42

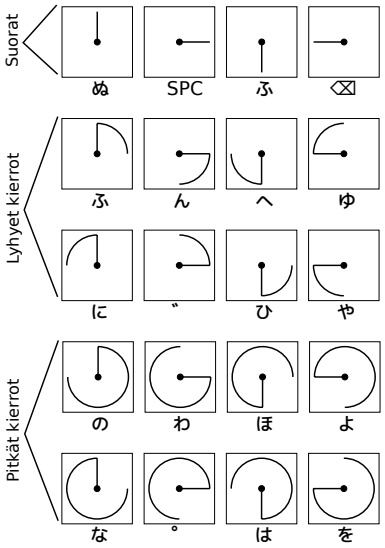
[61] O. Jacob Wobbrock. Measures of text entry performance. In *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufman, 2007. 16, 17, 18, 19, 20

[62] Thomas G. Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. *SIGCHI Bull.*, 18(4):189–192, 1987. 34

A Elekartat

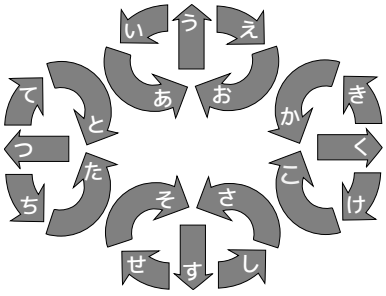


(a) Vasemman sauvaohjaimen eleet.

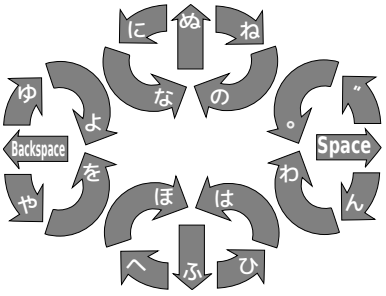


(b) Oikean sauvaohjaimen eleet.

Kuva A.1: Hiragana-tavuaakkosto.



(a) Vasemman sauvaohjaimen eleet.



(b) Oikean sauvaohjaimen eleet.

Kuva A.2: Vaihtoehtoinen visualisointi eleaakkostolle.

B Algoritmit

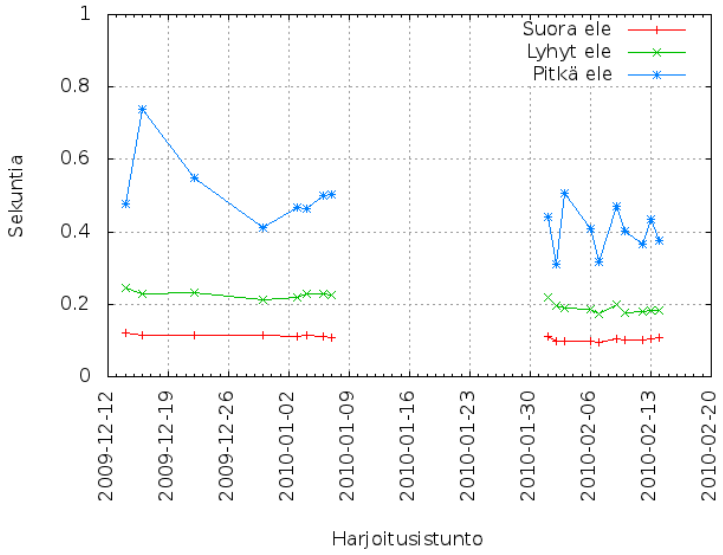
Listing B.1: Python-toteutus Bresenhamin algoritmista.

```
def bresenham(point1, point2):
    if point1 == point2:
        yield point1
        return

    x1, y1 = point1
    x2, y2 = point2
    is_steep = abs(y2 - y1) > abs(x2 - x1) # Slope > 1.

    if is_steep:
        x1, y1 = y1, x1
        x2, y2 = y2, x2

    if x1 > x2:
        x1, x2 = x2, x1
        y1, y2 = y2, y1

    dx = x2 - x1
    dy = abs(y2 - y1)
    err = 0.0
    derr = dy / dx
    y = y1
    ystep = 1 if y1 < y2 else -1

    for x in range(x1, x2 + 1): # x2 + 1 to yield point2 too.
        yield (y, x) if is_steep else (x, y)
        err += derr
        if err >= 0.5:
            y += ystep
            err -= 1.0
```

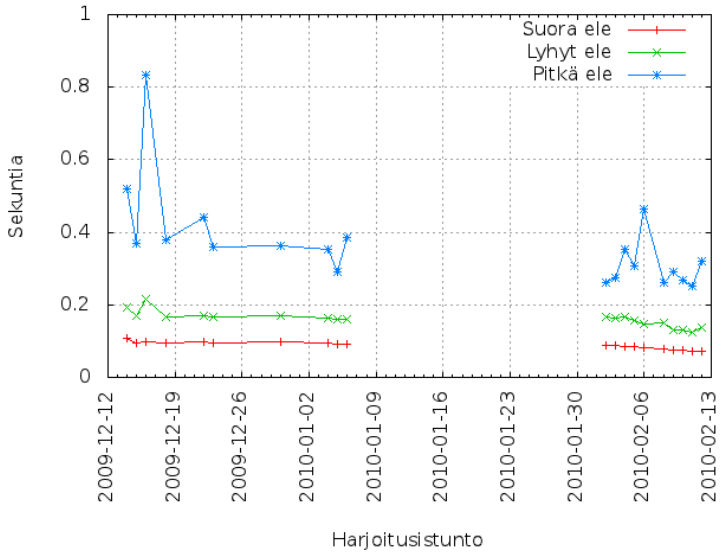
Listing B.2: Python-toteutus Levenshteinin algoritmista.

```
def msd(string1, string2):
    if len(string1) > len(string2):
        string1, string2 = string2, string1
    current = range(len(string1) + 1)
    for i in range(1, len(string2) + 1):
        previous, current = current, [i] + [0] * len(string1)
        for j in range(1, len(string1) + 1):
            add, delete = previous[j] + 1, current[j - 1] + 1
            change = previous[j - 1]
            if string1[j - 1] != string2[i - 1]:
                change = change + 1
            current[j] = min(add, delete, change)
    return current[len(string1)]
```

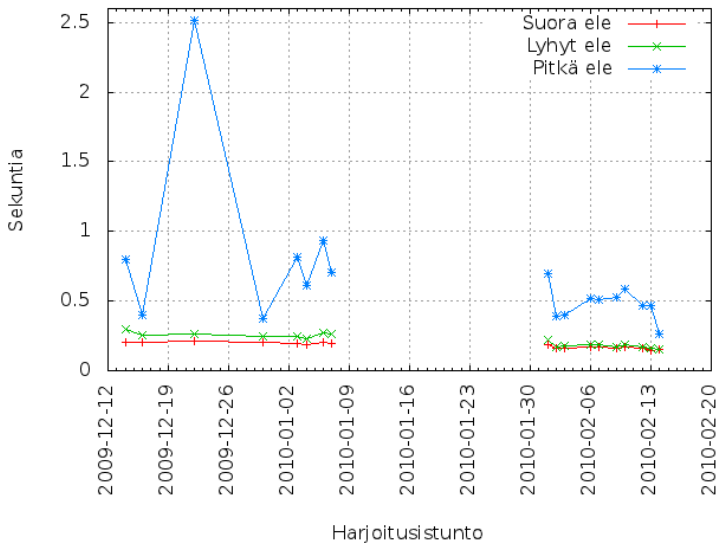
C Tulokset



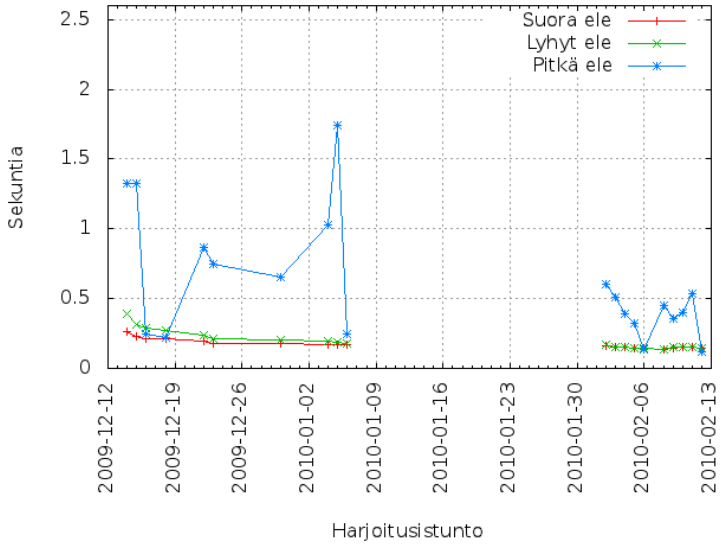
Kuva C.1: Koehenkilön 8 kolmannen käyttäjätestin keskimääräiset suoritusajat harjoitusistunnoittain jokaisen kolme eletyypin osalta.



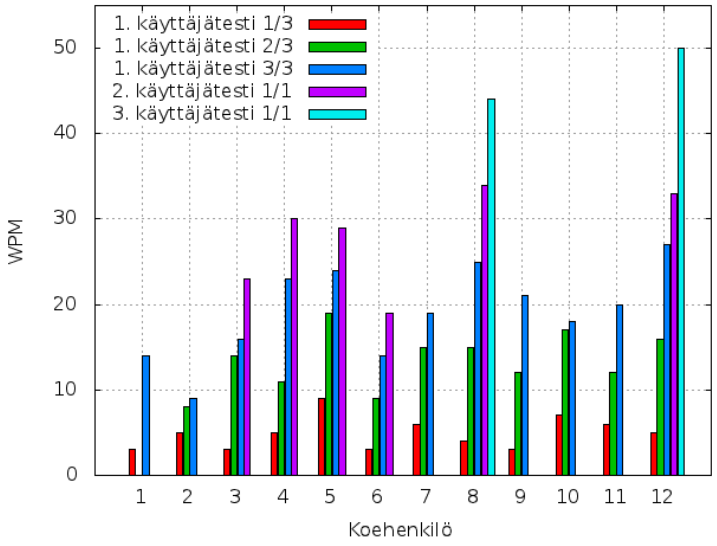
Kuva C.2: Koehenkilön 12 kolmannen käyttäjätestin keskimääräiset suoritusajat harjoitusistunnoittain jokaisen kolmen eletyypin osalta.



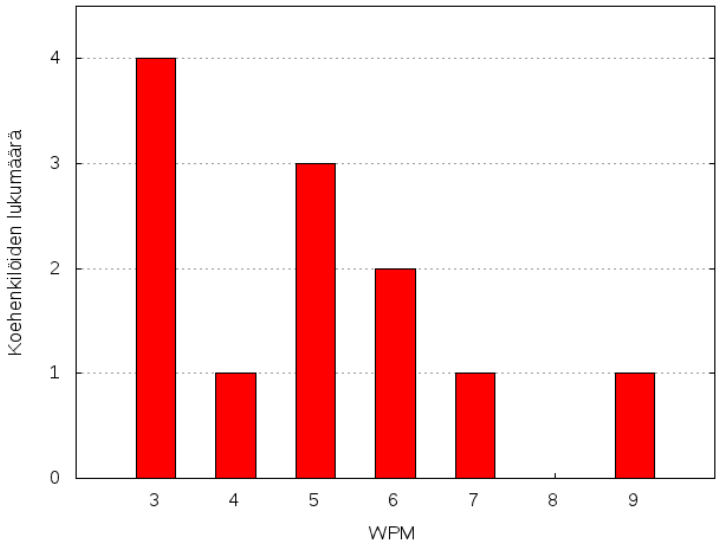
Kuva C.3: Koehenkilön 8 kolmannen käyttäjätestin keskimääräiset eleviiveet harjoitusistunnoittain jokaisen kolme eletyypin osalta.



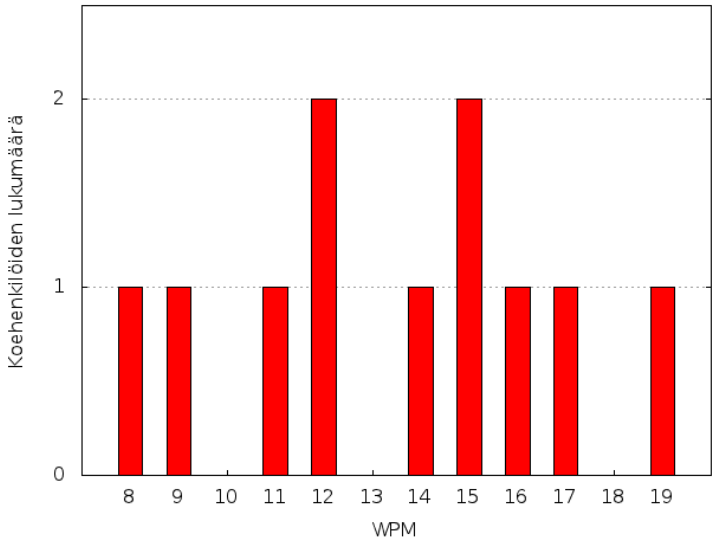
Kuva C.4: Koehenkilön 12 kolmannen käyttäjätestin keskimääräiset eleviiveet harjoitusistunnoittain jokaisen kolmen eletyypin osalta.



Kuva C.5: Huippukirjoitusnopeuksien kehitys koehenkilöittäin.



Kuva C.6: Ensimmäisen käyttäjätestin koehenkilöiden huippukirjoitusnopeuksien diskreetti jakauma ensimmäisen mittauksen jälkeen.



Kuva C.7: Ensimmäisen käyttäjätestin koehenkilöiden huippukirjoitusnopeuksien diskreetti jakauma toisen mittauksen jälkeen.