Dynaamista geometriaa Moodle-ympäristöön

STACK- ja JSXGraph-järjestelmien testaamista monimuotoisten kysymysten laatimiseksi

Itä-Suomen yliopisto
Luonnontieteiden ja
metsätieteiden tiedekunta
Fysiikan ja matematiikan laitos
Pro gradu -tutkielma
Kesäkuu 2017
Henri Tanskanen
Ohjaaja: Martti Pesonen

Tiivistelmä

Tämä ABACUS-projektiin liittyvä tutkielma on suurelta osin työraportti Itä-Suomen yliopiston fysiikan ja matematiikan laitoksella Moodle-oppimisympäristöön konstruoiduista STACK-kysymyksistä lukuvuoden 2016-2017 aikana. Työjakso sisälsi Moodle- ja STACK-järjestelmiin tutustumisen ja niiden käyttämistä opettajien ehdottamien kysymysten muodostamisessa. Dynaamista geometriaa sisältävien tehtävien laatiminen STACK-Moodle -kysymyksiksi muodostui merkittäväksi osaksi tutkimusta. Laitoksen toiminta painottuu opettajankoulutukseen, ja siksi käsitteiden visualiseen havainnollistamiseen kiinnitetään koulutuksessa enemmän huomiota kuin pelkkiin laskutehtäviin.

Työjakson aikana laadittiin useita kysymyssarjoja, enimmäkseen kursseille Matematiikan johdantokurssi ja Lineaarialgebra. Tietokoneavusteiseen arviointiin pystyvällä STACK-järjestelmällä luotiin monipuolisia Moodle-kysymyksiä, jotka usein sisälsivät myös satunnaisuutta. Dynaamiset kuviot laadittiin JSXGraph-ohjelmakirjastolla, ja niissä kokeiltiin myös kysymystyyppiä, jossa vastaus annetaan kuvion objekti(e)n asemaa muuttamalla. Aiemmin laitoksen opetuksessa ja tutkimuksissa hyödynnetyt dynaamisen geometrian JavaSketchpad-kuviot (JSP) pyrittiin myös nykyaikaistamaan ja tuomaan osaksi näitä tehtäviä. Tähän tarkoitukseen ohjelmoitiin kääntäjä, joka muuntaa JSP-koodit nykyisiin verkkoselaimiin paremmin sopivaan JSXGraph-muotoon. Lisäksi tutkittiin opiskelijoiden valmiuksia ratkaista STACK-järjestelmällä laadittuja tehtäviä. Samalla tarkasteltiin kuinka dynaamiset kuviot auttavat opiskelijoita matemaattisten käsitteiden oppimisessa.

Työjakson tulokset olivat pääosin myönteisiä. STACK-järjestelmä todettiin hyvin luotettavaksi ja monipuoliseksi työvälineeksi kysymysten laadintaan. Moodle-harjoitusten ja -tenttien teettämisen yhteydessä saatiin selkeä kuva niiden toimintavarmuudesta ja asioista, joihin kannattaa kiinnittää huomiota. Dynaamisten kuvioiden upottaminen kysymyksiin oli valmiiden kuvien ja kääntäjän ansiosta tehokasta; vanhat kuviot kääntyivät uusiin kysymyksiin hyvin luotettavasti ja nopeasti sitä mukaa kun kääntäjän toimintaa paranneltiin. Opiskelijoiden kokemuksia mitattiin tenttipalautteilla ja kurssien jälkeisellä kyselyllä. STACK-kysymysten hyödyntäminen kursseilla koettiin mielekkääksi. Tietokoneharjoitukset koettiin hyväksi lisäksi kurssin suorittamiseen, etenkin asioiden kertaamiseen. Dynaamisten kuvioiden merkitys oppimisen kannalta oli opiskelijoiden mielestä tärkeä.

Avainsanat: STACK, Moodle, JSXGraph, dynaaminen geometria, JavaSketchpad, Maxima, tietokoneavusteinen arviointi, tietokonealgebrajärjestelmä

Abstract

This Master thesis, related to ABACUS project, is largely a work report on constructing STACK questions into the Moodle learning environment at the Department of Physics and Mathematics of the University of Eastern Finland during the academic year 2016-2017. The work consisted of getting familiar with Moodle and STACK systems and using these systems for generating questions suggested by teachers. Constructing exercises including dynamic geometry into STACK-Moodle questions became a significant part of this piece of research and development project. The department is focused in teacher training, and therefore much attention is paid in visualizing mathematical concepts.

Several STACK question sequences were developed during the work period, mostly for courses Introduction to Mathematics and Linear Algebra. Versatile Moodle questions were constructed with the STACK system, which is capable in performing computer aided assessment. Often these questions included random elements. Dynamic sketches were created with the JSXGraph software, and a question type where the answer is given by changing the position of some objects in the sketch was also tested. Numerous dynamic JavaSketchpad sketches (JSP) that had been formerly used in teaching and research were modernized and reformulated to fit in the new materials. For this purpose, a compiler was programmed to convert JSP codes to JSXGraph form, which is more suitable for modern web browsers. Students' ability to solve exercises created with the STACK system was studied, and it was also examined how dynamic sketches help students learn mathematical concepts.

The results of the working period was positive. The STACK system was found to be a very reliable and versatile tool for constructing questions. A good understanding of the functionality of STACK questions was obtained. When making the students do these Moodle exercises and quizzes, a clear picture was gained about their reliability and issues you should pay attention to. Embedding dynamic figures into the questions became efficient with ready-made sketches and the programmed converter. The students' experiences were recorded from quiz feedback questions and a questionnaire after courses. The use of STACK questions on the courses was reported to be meaningful. Computer exercises were acknowledged as beneficial supplements in the courses, especially for rehearsing things. The significance of dynamic figures for learning was important for students.

Keywords: STACK, Moodle, JSXGraph, dynamic geometry, JavaSketchpad, Maxima, computer aided assessment, computer algebra system

Käytettyjä lyhenteitä

CAA Computer Aided Assessment (*Tietokoneavusteinen arviointi*)

CAS Computer Algebra System (*Tietokonealgebrajärjestelmä*)

DGE Dynamic Geometry Environment (*Dynaaminen geometriaympäristö*)

DOM Document Object Model (*Dokumenttioliomalli*)

JSP JavaSketchpad

STACK System for Teaching and Assessment using a Computer algebra Kernel

Sisältö

Kuvat

Listaukset

Esipuhe

Tietokoneen hyödyntäminen matematiikan opetuksessa ja havainnollistamisessa on paitsi kiinnostanut minua aina, myös ollut tarpeellinen osa oman matemaattisen osaamiseni kehitystä ajatellen. Itse asiassa kiinnostukseni matematiikkaan heräsi vasta lukion toisella luokalla, kun sain ensimmäisen graafisen laskimeni. Mahdollisuus piirtää kuvaajia laskimella, ja erilaisten symbolisten ja graafisten asioiden vaivaton hahmottelu laskimella, helpotti opiskeluani huomattavasti.

Tieteenalat, joita itse olen opiskellut, ovat matematiikan lisäksi pedagogiikka ja tietojenkäsittelytiede. Matematiikan ja tietotekniikan yhdistäminen on ollut minulle kiinnostuksen kohde jo opintoja valitessani. Matemaattisten rakenteiden visualisointi ja kokeilu dynaamisia kuvioita käyttäen on mielestäni oppimisen kannalta tärkeässä roolissa hyvin monen opiskelijan kohdalla.

Suurkiitos työni ohjaajalle Martti Pesoselle tuesta ja mielenkiintoisista haasteista kuluneen lukuvuoden aikana. Kiitän myös Antti Rasilaa ja Matti Harjulaa Aalto-yliopistosta käsikirjoituksen kommentoinnista ja hyödyllisistä vinkeistä. Lisäksi haluan kiittää perhettäni, erityisesti vaimoani Piiaa kärsivällisyydestä ja kannustamisesta opintojeni aikana.

Polvijärvellä 21.6.2017

Henri Tanskanen

1 Johdanto

Tässä tutkielmassa tarkastellaan matemaattisten STACK-tehtävien konstruoimista eri kurssien tarkoituksiin Itä-Suomen yliopiston Joensuun kampuksella lukuvuonna 2016-2017. Itä-Suomen yliopiston fysiikan ja matematiikan laitos on ollut alkuvuodesta 2016 mukana Aalto-yliopiston käynnistämässä ABACUS -projektissa, jonka pääasiallinen tarkoitus on tuottaa materiaalia kaikkien projektissa mukana olevien tahojen ja yhteistyökumppanien käyttöön (Rasila, 2016). Tutkielma perustuu lukuvuoden aikana tekemääni työhön, jossa muodostettiin useita matemaattisia STACK-kysymyssarjoja Moodle-oppimisympäristöön. Suuri osa konstruoiduista tehtävistä muotoiltiin vanhojen, aiemmin opetuksessa käytettyjen tehtävien tai ideoiden pohjalta STACK-tehtäviksi. Nämä tehtävät sisälsivät dynaamisia geometrisia kuvioita, joiden avulla oppimista on pyritty kehittämään Itä-Suomen yliopistossa aktiivisesti jo 2000-luvun alusta lähtien.

Tietotekniikan hyödyntäminen matematiikan opetuksen apuvälineenä on kehittynyt sitä mukaa kuin on päivittynyt käytettävä teknologia. Teknologian käyttö matematiikan opetuksessa on nykyään hyvin monipuolista sisältäen virtuaalisia oppimisympäristöjä, tietokonelaskentaa, tietokoneavusteista arviointia ja matemaattisten käsitteiden visualisointia tietokoneella. Näiden apuvälineiden didaktinen funktio on helpottaa niin opettajan työtä matematiikan opetuksessa kuin auttaa opiskelijaa näkemään ja kokeilemaan matemaattisia käsitteitä uudella tavalla. Teknologian käyttöön liittyy kuitenkin ongelmia, ja opiskelijoiden tekninen osaaminen voi asettaa heidät eriarvoiseen asemaan harjoituksia tehdessä. Opettajien on hallittava hyvin opetuksessa käytettävä tekniikka. Lisäksi järjestelmien täytyy olla toimintavarmoja.

Matematiikan opetuksen, oppimisen ja arvioinnin kehittämiseksi on käytetty moninaisia apuvälineitä aikojen saatossa. Tietokoneet erilaisine sovelluksineen ovat olleet mukana tässä kehityksessä niin kauan kuin tietokoneitakin on ollut. Vaikka ohjelmistojen ja tietokoneiden kehitys on ollut nopeaa viimeisten vuosikymmenten aikana, vuosituhannen alussa niiden hyödyntäminen matematiikan oppivälineenä oli vielä hyvin vähäistä (Ruthven ja Hennessy, 2002). STACK on matemaattisten aineiden järjestelmä, joka on pääasiassa rakennettu arvioimaan opiskelijoiden vastauksia monipuolisesti, antaa halutessa täsmällistä palautetta välittömästi ja mahdollistaa satunnaisuuden tehtävien sisällä.

Tietokoneet ovat olleet mukana laitoksen opetuksessa jo 1980-luvun loppupuolelta alkaen. Joillakin kursseilla tietokoneharjoituksien ja -tenttien tulokset ovat myös vaikuttaneet kurssin arvosanaan merkittävässä määrin. Virtuaaliset oppimisympäristöt ovat olleet laitoksen käytössä 2000-luvun alusta lähtien, aluksi WebCT ja myöhemmin Moodle, joka on nykyään käytössä oleva ympäristö. Nämä oppimisympäristöt ovat olleet säännöllisesti tutkimuskohteena tai alustana pienimuotoisille tutkimuksille. Laitoksen tarjoaman koulutuksen painoarvo on opettajankoulutuksessa, ja siksi matemaattisten käsitteiden visuaaliseen havainnollistamiseen kiinnitetään paljon huomiota. Esimerkiksi dynaamisten kuvioiden käyttöä käsitteiden hahmottamiseksi on tutkittu useissa projekteissa (Pesonen ym., 2002), (Pesonen ym., 2006). Tietokoneavusteista matematiikkaa on hyödynnetty laitoksen opetuksessa ja tutkielmissa säännöllisesti. Viimeisimpiä tutkielmia aiheeseen liittyen on kirjoittanut Muli (2014) tutkimuskohteenaan relaatiokäsitteen testaaminen Moodle-harjoitusten avulla kurssilla Matematiikan Johdantokurssi (2012). Tuolloin ei tehtävissä enää voitu käyttää Java-pohjaisia dynaamisia kuvioita.

Nykyään matemaattisten ohjelmistojen kirjo on laaja niin algebrallisia laskutoimituksia suorittavissa kuin myös geometrisia konstruktioita tarjoavissa sovelluksissa. Valikoimaa on runsaasti eritasoisille käyttäjäryhmille, myös yliopisto-opiskelijoille ja -opettajille. Dynaamisia sovel-

luksia ja harjoituksia voidaan käyttää paitsi havainnollistamaan matemaattisia ilmiöitä, myös suoraviivaiseen harjoitteluun tai osaamisen testaamiseen. Itä-Suomen yliopistossa käytetyistä tehtävistä suuri osa on ollut dynaamisia kuvioita hyödyntäviä. Tässä tutkielmassa tarkastellaan näiden tehtävien tehoa matemaattisten rakenteiden hahmottamisessa sekä uudelleenkonstruoimista uusilla sovelluksilla ja STACK-järjestelmään soviteltuna.

Vaikka tietokonepohjaisten sovellusten hyödyntäminen opetuksessa tuntuu tänä päivänä itsestäänselvältä, liittyy siihen kuitenkin monia haasteita ja huomioitavia asioita. Ongelmia voi ilmetä esimerkiksi sovellusten toimivuudessa ja opiskelijoiden tietoteknisissä taidoissa. Huomioon on otettava myös opiskelijan kognitiiviset taidot, tietokoneen vuorovaikutus ja pedagoginen näkökulma. Myös itse opetustilanteeseen liittyy paljon kysymyksiä, kun opetusvastuu siirtyy osittain opettajalta tietokonesovelluksen hoidettavaksi. Tutkielmassa tarkastellaan myös jossain määrin näitä ongelmia yleisellä tasolla.

Myös opettajien osaamisessa saattaa olla suuria eroja. Joillekin opettajille tietokonepohjaisten harjoitusten laatiminen ja käyttö on tuttua, mutta joillekin haastavampaa. STACK-järjestelmän rakentamisen eräs periaatteellinen lähtökohta on ollut tehdä siitä mahdollisimman käyttäjäystävällinen niin opiskelijoille kuin myös tehtävien laatijoille ja opettajille. Tältä osin toimin itsekin tarkastelukohteena tässä tutkimuksessa, sillä aloittaessani STACK-tehtävien teon en ollut ennen kuullutkaan kyseisestä järjestelmästä. Tutkimuksen loppupuolella havainnollistan hieman kuinka paljon aikaa itselläni meni järjestelmän oppimiseen miltäkin osin.

Työni aikana hyödynsin myös useita muita sovelluksia, jotta dynaamiset tehtävät saatiin muodostettua. Aikaisemmin JavaSketchpad-kielellä tehtyihin dynaamisiin kuvioihin tahdottiin päivitystä uudemmalle sovellusalustalle, mistä lopulta kehittyi iso projekti käännösohjelman rakentamiseksi. Kääntäjä valmistui vuoden 2016 loppupuolella, mutta sen kehitys on jatkunut työni ohella. Kääntäjällä ajetut JavaSketchpad-koodit muuntuvat JSXGraph-muotoon, joka on JavaScript-pohjainen geometrian ja visuaalisen matematiikan esitystyökalu web-selaimille. Tutkielmassani kerron myös tästä kääntäjän kehittämisestä ja käytöstä.

2 Teoriaa ja käsitteitä

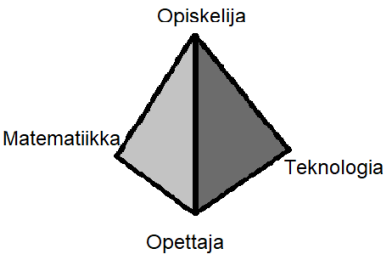
Tämä tutkimus keskittyy oleellisesti työskentelyyni STACK tehtävien laatimiseksi ja päivittämiseksi. Keskeistä oli STACK-järjestelmän käytön oppiminen, tehtävien kehittäminen ja aikaisemmin laadittujen tehtävien muokkaaminen STACK-järjestelmälle. Työn aikana myös matematiikan visuaalinen esittäminen, etenkin matemaattisten käsitteiden dynaaminen havainnollistaminen tehtävissä muodostui tärkeäksi tarkastelukohteeksi. Työtä tuki paitsi aiheeseen liittyvä kirjallisuus myös yliopiston henkilökunnan tietämys ja osaaminen.

2.1 Tietokoneavusteinen matematiikka

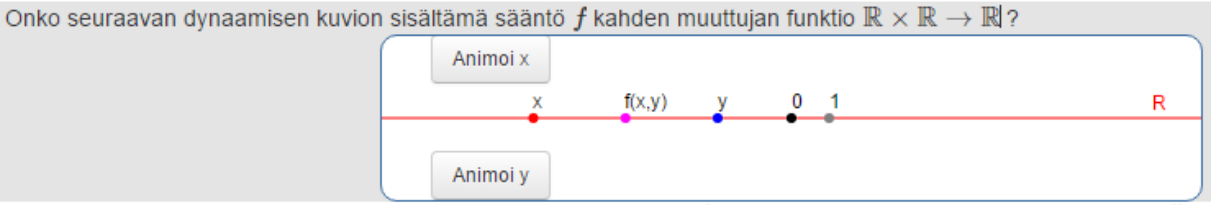
Matematiikka poikkeaa muista tieteenaloista loogisuutensa ja symbolisen rakenteensa vuoksi (Burn ym., 1998). Matemaattisen ajattelun tueksi on kehitelty useita apuvälineitä, jotka helpottavat matemattisten käsitteiden havainnollistamista ja tukevat oppimista. Eräitä esimerkkejä tällaisista apuvälineistä ovat viivoitin, helmitaulu, laskin sekä merkittävä symbolinen työkalu, karteesinen koordinaattijärjestelmä. Myös erityisiä apuvälineitä on kehitetty tukemaan matemaattista opetusta ja oppimista, joista yhtenä esimerkkinä Sutherland (2006) mainitsee symbolisen lukusuoran. Tietokoneet ovat luonnollisesti muodostuneet erittäin merkittäviksi apuvälineiksi runsaine sovelluksineen ja käyttötarkoituksineen. Vuosituhannen alusta niin teknologinen kehitys kuin ihmisten kyky hyödyntää teknologiaa on ollut erittäin nopeaa (Moreno-Armella ym., 2008).

Haapasalo (2007) listaa työvälineet, joita matematiikan opetuksessa nykyään hyödynnetään. Listalta löytyvät esimerkiksi symbolisen laskennan työkalut, dynaamisen geometrian työkalut, piirto-ohjelmat, hakukoneet ja verkon yli hyödynnettävät oppimisympäristöt. Kaikki nämä työkalut antavat matematiikan oppimiseen aivan uuden näkökulman; oppiminen ei ole sidottu aikaan, paikkaan eikä formaaliin muotoon (Haapasalo, 2007). Oppiminen voi jopa siirtyä teknologian avustamana opiskelijoiden vapaa-ajalle ja antaa heille mahdollisuuden kokeilla opetettavia käsitteitä vapaammin. Teknologian vaikutusta opetukseen on usein kuvattu (Hollebrands, 2016) didaktisella tetraedrilla (Kuva 1), missä teknologia on osa opiskelijan, opettajan ja matemaattisen osaamisen välistä vuorovaikutusta.

Tietokonetta voidaan käyttää opettamiseen ja oppimiseen monella tavalla. Periaatteessa kaikki tavat, joilla tietokoneita hyödynnetään tutkimuksessa voidaan käyttää myös matematiikan oppimis- ja opetustarkoituksiin (Dubinsky ja Tall, 1991). Esimerkiksi erilaiset ohjelmistot ovat hyvä tapa tarjota opiskelijoille ympäristö, jossa tutkia ja kokeilla ideoita. Oppimisympäristöjen ja sovellusten luominen ja hallinnointi avaa uusia mahdollisuuksia hyödyntää tietokoneita matematiikan opetuksessa, kun ympäristöt voidaan rakentaa täsmällisesti opettajan ja opiskelijan tarpeisiin (Wiest, 2001). Esimerkkinä tästä on kuvan 2 kysymys kurssilta Matematiikan johdantokurssi, jossa opiskelijan täytyy tutkia miten funktion arvo $f(x, y)$ käyttäytyy, kun muuttujien x ja y arvoja muutetaan.



Kuva 1: Didaktinen tetraedri



Kuva 2: Funktio-oppimodulin kysymys kurssilta Matematiikan johdantokurssi 2016

Kuvan 2 tehtävä on hyvä esimerkki eräästä merkittävästä teknologian tuomasta lisäedusta matematiikan oppimisessa; *virtualisointi*. Tietokoneen sovelluksella on mahdollista luoda eräänlainen uusi (virtuaali)todellisuus, jota ei muulla keinoin voida rakentaa (Borovik, 2017). Usein puhuttaessa matemaattisista objekteista puhumme asioista, joita ei todellisuudessa ole konkreettisesti nähtävillä, mutta tietokoneen sovelluksella voimme simuloida näitä objekteja.

Dubinsky ja Tall (1991) sanovat, että yksilöiden matemaattisen ymmärryksen arviointi ja hahmottaminen ei ole aivan suoraviivaista. Opiskelijat saattavat antaa kysymyksiin oikean vastauksen vääristä syistä ja päinvastoin. Usein opiskelijoiden virheet syntyvät aiemmin opitun asian soveltamisesta väärällä tavalla tai jopa väärässä asiayhteydessä. Näin ollen voisikin olla oppimisen kannalta parempi, jos opiskelijat rakentavat itse tietämyksensä uudesta asiasta ohjauksen, hahmottelun tai jopa simuloinnin avulla. Tällöin tietokone ja oikein kohdennettu sovellus ja sen käyttö tukevat oppimista ja jopa auttavat opiskelijaa rakentamaan tietämyksensä asiasta itse, kun ideoita voi kokeilla, muokata ja saada suoraa palautetta. (Dubinsky ja Tall, 1991).

Vaikka ongelmien ratkaiseminen tietokoneella onkin hyödyllistä monessa mielessä, voi tietokoneen käyttäminen oppimistarkoituksissa asettaa haasteita. Kun käytetään matemaattisia tietokonesovelluksia tarvitaan usein myös taitoa käyttää kyseistä ohjelmaa. Kun opiskelija keskittyy uuden matemaattisen rakenteen oppimiseen, saattaa häneltä kulua suhteellisen paljon resursseja myös itse sovelluksen ja ohjelmointikielen oppimiseen. (Grabmeier ym., 2003).

Osaamisen tason suuri vaihtelevuus opiskelijoiden välillä voi muodostua haasteeksi käytettäessä tietotekniikkaa opiskelun tukena (Wiest, 2001). Wiest (2001) toteaa kuitenkin, että tasoerot opiskelijoiden osaamistasoissa ovat jo vuosituhannen alussa olleet selvästi kaventumassa teknologian yleistyessä ja tullessa kaikkien saataville. Kor ja Chuah (2014) ovat tutkineet kuinka opiskelijat, joille tietokonesovelluksilla suoritettava matematiikan opiskelu on ennestään vierasta, suoriutuivat Maplella[1] tehdyistä harjoituksista työarkin avulla, ja kuinka suuren kognitiivisen taakan tällainen oppimismenettely asetti. Kor ja Chuah (2014) havaitsivat, ettei tietokoneavusteinen opetus työarkkia seuraamalla lisännyt kognitiivista kuormitusta, vaikka tekniikka itsessään oli uutta eikä sitä juurikaan opetettu. Lisäksi kiinnostus tekniikan hyödyntämistä kohtaan oppimisessa oli korkea.

Myös opettajien osaaminen tai ajanpuute opetella tarvittavien sovellusten käyttö voi olla haaste. Tämä ongelma korostuu, jos sovellusten käyttö vaatii paljon etukäteisopiskelua (Haapasalo, 2007). Heid ja Edwards (2001) toteavat raportissaan, että opettajien kognitiiviset taidot ja metodit tietotekniikan opetuskäytössä siirtyvät suoraan opiskelijoiden tapaan hyödyntää tietotekniikkaa apuvälineenä.

[1] https://www.maplesoft.com/products/maple/

2.2 Visuaalisuuden merkitys matematiikan oppimisessa

Käsitekuvalla tarkoitetaan jonkin käsitteen visuaalista hahmottamista ilman käsitteen konkreettista havainnointia (Vinner, 1991). Kun näemme tai kuulemme tutun käsitteen, käsitekuva asiasta tulee mieleemme. Käsitekuva voi olla visuaalinen esitysmuoto tai mentaalinen kuva asiasta. Voimme kuvailla käsitteen tämän jälkeen sanoin, mutta Vinner (1991) huomauttaa, että tämä sanallinen kuvaus ei tavallisimmin ole asia, joka mieleemme tuli ensin. Esimerkiksi kun kuulemme sanan ”pöytä”, kuva tietystä pöydästä voi tulla mieleemme. Samoin kuullessamme sanan ”funktio”, joku saattaa hahmotella mielessään ilmaisun $f(x) = y$ tai jonkin funktion kuvaajan.

Visualisointia ei voida väheksyä uusien taitojen oppimisessa ja ratkaistaessa ongelmia (Grabmeier ym., 2003). Visuaalinen hahmottaminen aktivoi uudenlaisen ymmärtämisen tason ja auttaa selittämään ongelmaa. Hahmottamisella voidaan muuttaa abstrakti käsite vähemmän abstraktiin muotoon, jolloin asian ymmärtäminen saattaa helpottua. Tämä auttaa niin opiskelijaa kuin opettajaa. Varsinkin opiskelijat, joilla on vaikeuksia oppia abstrakteja käsitteitä, visualisointi voi olla merkittävä tekijä osaamisen kehityksessä. Visualisoinnin merkityksestä kertoo myös se, että älyllistä matematiikkaa suorittaessaan ihminen havainnollistaa mielessään sen apuvälineen, jota kyseiseen ongelmaan tarvitaan (Sutherland, 2006). Esimerkiksi ajatellessa jotakin tason pistettä se usein visualisoidaan mielessä karteesiseen koordinaatistoon.

Analyyttinen matemaattisten käsitteiden ajattelu on usein hyvin tarkka ja yksityiskohtainen menetelmä. Visualisointi sen sijaan voi muodostaa ongelman kokonaisvaltaisemmat piirteet konkretisoimalla sitä, toimien samalla eräänlaisena karttana, joka osoittaa suunnan päättelyprosessille (Viholainen, 2008). Tällainen ajattelu edellyttää kuitenkin sitä, että visuaalisen ja symbolisen esitysmuodon välinen yhteys ymmärretään syvällisesti (Viholainen, 2008).

Matemaatikoilla on yleensä kyky hyödyntää visualisointia tehokkaasti (Nishizawa ym., 2012). Matemaatikot kykenevät yhdistämään visuaalisen ja analyyttisen vaiheen niin, että ne toimivat läheisessä vuorovaikutuksessa toistensa kanssa. Opiskelijoiden kohdalla asia ei kuitenkaan ole näin. Monien opiskelijoiden on vaikeaa analysoida matemaattisten käsitteiden visuaalisia esitysmuotoja, ja siksi heillä on vaikeuksia hyödyntää niitä ongelmanratkaisussa (Stylianou ja Dubinsky, 1999). Usein opiskelijat ajattelevat, että visuaalinen hahmottaminen ja käsitteiden järkeily ei ole matemaattisesti sallittu menetelmä (Stylianou ja Dubinsky, 1999). Visuaalisen esitysmuodon käyttö on tehokas käsitteen hahmottamisen apuväline, ja opiskelijan tulisi oppia käyttämään hyväksi sekä visuaalista että symbolista esitysmuotoa (Rasila, 2013). Siksi onkin tärkeää, että sekä visuaalinen että analyyttinen päättely ja niiden välinen yhteys korostuvat matemaattisessa ajattelussa ja opetuksessa (Nishizawa ym., 2012).

2.3 Dynaaminen geometria

Tietokoneella voidaan luoda dynaaminen geometria-ympäristö jollakin tarkoitukseen laaditulla geometriaohjelmalla. Sovelluksella rakennetaan jokin geometrinen konstruktio, jonka jo(i)takin objekteja voidaan sormella tai tietokoneen hiirellä siirtää siten, että määrätyt säännönmukaisuudet kuviossa säilyvät (Hollebrands, 2016). Tämä hiirellä *siirtäminen (dragging)* on dynaamisen geometrian määrittelevin ominaisuus (Sinclair ja Yurita, 2008).

Moreno-Armella ym. (2008) ovat esitelleet viiden tason mallin siitä kuinka matemaattisten käsitteiden visuaalinen tarkastelu on kehittynyt:

- *Taso 1. Staattinen liikkumaton:* Kirjoitukset, esimerkiksi painettu teksti oppikirjoissa.
- *Taso 2. Staattinen kinesteettinen/esteettinen:* Uudelleen käytettävä alusta; esimerkiksi liitutaulu, josta voidaan pyyhkiä osa tai kaikki kirjoitukset pois ja kirjoittaa uudelleen.
- *Taso3. Staattinen laskennallinen:* Alusta, joka osaa esittää käyttäjän syötteet ja niiden muutokset; esimerkiksi laskin, joka kuvaa käyttäjän syötteitä ja tulosta staattisilla ilmauksilla.
- *Taso 4. Diskreetisti dynaaminen:* Laskennalliset tulokset mukautuvat käyttäjän syötteen mukaan; esimerkiksi laskentakaavio, joka muodostaa käyttäjän syötteiden arvoista kuvaajan, joka mukautuu dynaamisesti käyttäjän muuttaessa joitakin arvoja.
- *Taso 5. Jatkuva dynaaminen:* Tämä taso kehittyy edellisestä siten, että käyttäjän fyysistä tekemistä ja vuorovaikutusta seurataan keskeytymättä. Esimerkiksi pisteen siirtäminen hiirellä jossakin dynaamisessa kuviossa muuttaa välittömästi jollain lailla joitain kuvion arvoja.

Voidaan todeta, että jokaista tasoa myös edelleen sovelletaan matematiikan päivittäisessä opetuksessa. Uusi kehityksen taso ei siis ole tehnyt edellisen tason käyttöä tarpeettomaksi, vaan on tullut tukemaan sitä. Nykyisin on saatavilla paljon erilaisia dynaamisen geometrian sovelluksia, mutta kaikki ne toimivat samalla periaatteella: perusobjektit (piste, suora ja ympyrä), yleiset rakennustyökalut (samansuuntaisuus, kohtisuoruus yms.), siirrot, mittaukset ja objektien siirtäminen hiirellä (Hollebrands, 2016).

Mahdollisuus muodostaa dynaamisesti muuttuvia interaktiivisia kuvioita, objekteja ja järjestelmiä on muuttanut matematiikan opetusta merkittävästi. Moreno-Armella ym. (2008) korostaa dynaamisen geometrian merkitystä sen tarjoaman välittömän palautteen takia, jota voidaan hyödyntää paitsi oppimisessa myös teorioiden todistamisessa. Dynaamisen geometrian soveltaminen opetuksessa voi kuitenkin osoittautua haastavaksi (Laborde, 2002). Laborde on tutkinut, kuinka opettajat laativat dynaamista geometriaa sisältäviä tietokonepohjaisia tehtäviä. Hän havaitsi, että opettajat käyttivät dynaamisen geometrian sovelluksia laatiessaan normaaleja tehtäviä tai sovellukset toimivat vain visuaalisina esitystyökaluina. Pitkäkestoisemman käytön tuloksena opettajat alkoivat kuitenkin konstruoida tehtäviä, joita ei voisi tehdä ilman teknisiä ympäristöjä. Saman havaitsivat myös Sinclair ja Yurita (2008) tutkimuksessaan, jossa he tarkastelivat kahden lukio-opettajan opetusmetodien muutosta siirryttäessä staattisten kuvioiden käytöstä dynaamisiin. Siirtymisen aikana opettajien kyky tuottaa tarkoituksenmukaisia dynaamisia kuvioita kehittyi selvästi.

Teknologia muuttaa matemaattista ajattelua, ja tehtävien laatijoiden täytyy huolella miettiä matematiikan, teknologian, opettamisen ja opiskelun välistä suhdetta. On siis syytä miettiä, milloin dynaamisen geometrian käyttäminen tehtävissä on oppimisen kannalta järkevää. Opiskelijat eivät välttämättä ole tottuneet dynaamisiin kuvioihin, jolloin opiskelijan huomio varsinaisen tehtävän sijaan saattaakin kiinnittyä käytettyyn teknologiaan tai ohjelmistoon. Hollebrands (2016) ja Pesonen ym. (2006) toteavat tutkimuksessaan, että onkin hyödyllistä tarjota ensin tehtäviä, jotka tekevät käytettävän oppiympäristön tutuksi, ja vasta myöhemmin tehtäviä, jotka syventyvät teoria-asioihin. Dynaamisilla geometria-kuvioilla voidaan myös näyttää arvoja ja esityksiä hyvin tarkasti, jolloin on mahdollista etteivät opiskelijat välttämättä vaivaudu osoittamaan otaksumaa todeksi, vaan ennemmin tutkivat kuvion avulla sen ilmiön syitä. Oppimisvälineenä dynaamiset kuviot ovat myös paljon aikaa vieviä, varsinkin jos tämän tyyppinen työskentely on opiskelijoille uutta (Pesonen ym., 2006).

2.4 Tietokoneavusteinen arviointi

Arvioinnilla yleensä halutaan selvittää se, kuinka hyvin opiskelija hallitsee oppimansa asiat. Arviointiprosessissa opettaja muodostaa arvion opiskelijan työstä ja/tai tuloksesta sekä osaamisesta suhteessa tavoitteisiin ja asettaa tälle osaamiselle jonkin arvon, joka voi olla sanallinen tai vaikkapa numeerinen. Arviointi on kaiken oppimisen ja opetuksen avaintavoite. Arvioinnilla voidaan usein myös tutkia kuinka hyvin opettaminen on onnistunut. (Sangwin, 2015). Useimmiten arviointi on myös oppimiseen motivoiva tekijä. Kuten opetuksessa, myös arviointitehtävässä opettaja voidaan Sangwinin mukaan osittain korvata muilla keinoin.

Arviointia voidaan hyödyntää monin eri tavoin. Sangwin (2012) on luokitellut arviointitavat neljään tyyliin sen mukaan, mitä arvioinnilla halutaan saavuttaa tai tarkastella. Diagnostinen arviointi *(Diagnostic assessment)* kuvaa sitä, kuinka hyvin oppija hallitsee aiemmin oppimansa asiakokonaisuuden soveltamista ja edelleen kehittämistä. Diagnostisessa arvioinnissa palautteena annetaan melko tarkka kuvaus siitä, mihin osaamisalueeseen kannattaa kiinnittää huomiota aikaisemmin opitussa ja tulevassa oppimistyössä.

Formatiivinen arviointi *(formative assessment)* tukee paitsi opiskelijan, myös opettajan oppimista. Palaute on jatkuvaa (Majander ja Rasila, 2011) esimerkiksi kurssin aikana (Majander, 2010) ja opiskelijan kehittymistä seurataan aktiivisesti antamalla vaikkapa suullista palautetta. Palaute on siis hyvin henkilökohtaista, ja opiskelijan edistymistä ja vastauksia voidaan verrata esimerkiksi valmiisiin mallivastauksiin. Opettaja saa palautetta omasta työstään opiskelijan kehittymistä seuraamalla.

Summatiivinen arviointi *(summative assessment)* kertoo opiskelijan suoriutumisen määrän. Sillä voidaan siis kuvata opiskelijan edistymistä vaikkapa kurssin aikana, ja se voi olla esimerkiksi prosenttiluku suoritetuista opinnoista. Summatiivisessa arvioinnissa eri suoritusten painoarvoa voidaan korostaa ja esimerkiksi lopputentin painoarvoa voidaan pitää suurempana kuin kurssin aikana tehtyjen harjoitusten painoarvoa. Summatiivisen arvioinnin avulla opiskelijan oppimisen tahti, ja sitä kautta taitotaso pyritään varmistamaan. Laadun arvioinnilla *(Evaluative assessment)* mitataan opetuksen ja koulutuksen toimivuutta ja tehokkuutta.

Mitkä ovat hyvät lähtökohdat matemaattisen osaamisen arvioinnille? Sangwin (2012) esittää, että matemaattisten taitojen arvioiminen heijastuu kykyyn soveltaa osaamistaan käytännössä. Matemaattinen osaaminen näkyy kyvyissä ratkaista matemaattisia ongelmia, jolloin arviointia voitaisiin suorittaa esimerkiksi siitä, ymmärtääkö opiskelija yhdistää tarvittavan tiedon ongelmaan, ymmärtääkö hän perusteet ja lähtökohdat sen taustalla, sekä osaako hän soveltaa sitä ongelmaan (Sangwin, 2012). Matematiikassa myös ratkaisujen perustelu ja tarkkuus ovat hyvin merkittäviä osatekijöitä.

Tietokoneavusteisella arvioinnilla (*Computer Aided Assessment*, CAA) tarkoitetaan tietokoneen suorittamaa opiskelijan osaamisen arviointia ja testaamista. Tietokoneavusteista arviointia on käytetty jo useiden vuosikymmenien ajan, aina 1960-luvulta saakka (Sangwin, 2015). Tietokoneavusteista arviointia on alettu käyttää ensisijaisesti tehostamaan ja automatisoimaan arviointiprosessia ja siihen käytettyä aikaa. Tietokoneavusteisessa arvioinnissa opiskelijan vastausta verrataan useimmiten johonkin valmiiksi asetettuun tai laskettuun mallivastaukseen.

Tietokoneavusteinen arviointi vaatii sekä tehtävien laatijalta että suorittajalta tarkkuutta. Vaikka oppimisympäristö sisältää kehittyneemmän tavan tarkistaa vastaus symbolisen laskennan järjes-

telmää käyttäen, täytyy niin mallivastauksen kuin opiskelijan syötteen olla syntaktisesti oikein. Tehtäviä laatiessani tämä tarkoitti useita testauksia tehtävillä mahdollisten virheiden löytämiseksi. Usein ohjeet vastauksen syöttämiselle ovat tarpeelliset, jotta arviointia ei tehdä väärin virheellisesti annetun syötteen takia. Tästä näkökulmasta katsottuna tietokoneavusteisella arvioinnilla ei välttämättä saavuteta ajallista etua verrattuna perinteisten tehtävien laatimiseen ja arviointiin (Sangwin, 2012).

Tietokoneavusteisen arvioinnin aikana palaute opiskelijalle tulee välittömästi, kun taas perinteisessä arvioinnissa palautteen saamiseen voi kulua päiviä, jopa viikkoja. Tässä ajassa opiskelija on voinut unohtaa kyseisen tehtävän yksityiskohdat. Tässä suhteessa tietokoneavusteinen arviointi on selvästi tehokkaampi väline palautteen antoon ja tukee tehtäväkohtaisen asian oppimista. Lisäksi arviointia voidaan tehdä nopeasta hyvinkin suurelle opiskelijamäärälle mikä säästää tarkistamiseen käytettyä aikaa. (Sangwin, 2012).

Tietokoneavusteinen arviointi voidaan suorittaa yksinkertaisimmin monivalintatehtävillä. Tällöin on selkeästi asetettu oikeat ja väärät valinnat, ja tietokoneavusteinen arviointi/tarkastus ja vastauksesta annettava palaute voidaan määritellä selkeästi tehtävän sisällä. Keskitymme tässä tutkimuksessa kuitenkin tehtävätyyppeihin, joissa vastaus syötetään algebrallisena lausekkeena tai kuvan objekteja oikein säätämällä. Tällöin käytämme ratkaisun oikeellisuuden tarkastamiseen *tietokonealgebrajärjestelmää*, jota käsitellään seuraavassa kappaleessa.

2.5 Tietokonealgebrajärjestelmä

Tietokonealgebrajärjestelmä (*Computer Algebra System*) on sovellus, jolla voidaan käsitellä ja muokata matemaattisia objekteja esittäviä tietorakenteita (Sangwin, 2012). Tietokonealgebrajärjestelmät ovat kehittyneet voimakkaasti tietotekniikan kehittymisen rinnalla yhdistäen algoritmiikan ja abstraktin algebran tietojenkäsittelyn metodeiksi ja tarjoten työvälineen sovelletun matematiikan ja tietojenkäsittelyn välimaastoon (Grabmeier ym., 2003). Tietokonealgebrajärjestelmien toiminta perustuu matematiikan algoritmilliseen ja loogiseen laskennan suorittamiseen (Grabmeier ym., 2003).

Merkittävimmät edut tietokonealgebrajärjestelmien käytössä ovat tarkkuuden parantaminen ja nopeus (Grabmeier ym., 2003). Järjestelmiä voidaan parhaillaan käyttää matemaattiseen kokeiluun ja testaamiseen sekä tieteelliseen laskentaan (Grabmeier ym., 2003). Grabmeier ym. (2003) sanoo, että tietokonealgebrajärjestelmät voivat kehittää opiskelijoiden ymmärrystä matemaattisten menetelmien käytöstä. Tämän lisäksi tietokonelaskennan sovelluksia voidaan hyvin käyttää myös tietokoneavusteiseen arviointiin (Sangwin, 2012). Tietokonealgebralla arviointia voidaan suorittaa matemaattisesta symboli-ilmauksesta, jolloin riittää, että opiskelijan syöte on ekvivalentti vaaditun vastauksen kanssa. Syötteen ei siis tarvitse symbolisesti vastata vaadittua ratkaisua, sillä laskentasovellus ymmärtää ovatko kaksi vastausta merkitykseltään samat.

Merkittävä hyöty tietokonealgebrajärjestelmien käytöstä oppimisympäristöissä on nimenomaan niihin rakennettu runsas määrä hyödyllisiä funktioita (Sangwin, 2012). Kysymyksistä voidaan luoda monipuolisia, ja ne voivat sisältää satunnaisuutta sekä omia funktioita vastausten tarkistamista varten. Järjestelmien käyttöön liittyvä kriittinen haaste on käyttäjän osaaminen. Vaikka monet laskentajärjestelmistä onkin rakennettu käyttäjäystävällisiksi, aikaa kuluu niiden tehokkaan hallinnan oppimiseen.

3 Tietoa ohjelmistoista

Tässä luvussa tarkastellaan kysymysten muodostamiseen tarvittavien sovellusten ominaisuuksia. STACK-kysymysten laadinta vaatii jonkin verran teknistä osaamista. Monet työjakson aikana konstruoiduista kysymyksistä vaativat usean järjestelmän käyttöä. Aikaisemmin laitoksella hyödynnetyt dynaamisen geometrian kuviot haluttiin muuntaa nykyaikaisempaan JSXGraph-muotoon. Alaluvuissa 3.1 - 3 esitellään tietokonealgebrajärjestelmä Maxima, ohjelmointikieli JavaScript ja dynaamisen geometrian sovellus JSXGraph. Alaluvuissa 3.4 - 5 esitellään dynaamisen geometrian sovellus JavaSketchpad ja käännössovellus, jolla JavaSketchpad-kuviot käännettiin JSXGraph-muotoon. Alaluvussa 3.6 tarkastellaan itse Moodle alustaa.

3.1 Maxima

Kuva 3: Maxima logo

Maxima on Lisp kielellä ohjelmoitu tietokonealgebrajärjestelmä. Systeemin kehitystyö alkoi vuonna 1968 ja sitä kehitettiin aktiivisesti vuoteen 1982 saakka Massachusettsin teknillisessä korkeakoulussa (MIT) (Grabmeier ym., 2003). Järjestelmä kulki tuolloin nimellä DOE Macsyma. Vuodesta 1982 vuoteen 2000 sovellusta ja sen kehittämistä piti yllä Texasin yliopiston professori William F. Schelter (Dautermann, 2016). Vuoden 1995 jälkeen Maxima on kehittynyt paljon käyttöliittymältään ja toiminnoiltaan (Grabmeier ym., 2003). Vuonna 2000 Schelter aloitti Maxima-projektin avoimen lähdekoodin kehittäjien yhteisössä nimeltään Sourceforge[2], mistä löytyy myös kattava Maximan manuaali[3] (Dautermann, 2016). Siitä lähtien sovellus on tunnettu nimellä Maxima. Maxima on STACK-järjestelmän taustalla toimiva tietokonelaskennan ja tietokoneavusteisen arvioinnin järjestelmä (Sangwin, 2012).

Maxima sisältää runsaan määrän erilaisiin laskentoihin liittyviä ominaisuuksia. Maximassa on käytössä useita erilaisia tietotyyppejä ja niitä voidaan käyttää melko vapaasti listoissa tai vaikkapa yhtälöissä. Esimerkiksi lista voi muodostua listoista, jotka sisältävät useaa eri tietotyyppiä kuten ohjelmointikielissä. Järjestelmällä voidaan suorittaa laskentaa käyttäen monia laskutoimituksia ja valmiita funktioita. Algebralliset laskutoimitukset ja yhtälöiden ratkaisualgoritmit toimivat ohjelmassa tehokkaasti. Voimme esimerkiksi määritellä matriisin komennolla

```
m1: matrix([2,5,11],[4,-5,3],[6,-7,3]);
```

ja laskea sen determinantin komennolla

```
det_m1: determinant(m1);
```

Maximassa on mahdollista käyttää myös ehtolauseita ohjelmointikielen tapaan, esimerkiksi

```
lin_rton: if det_m1 = 0 then false else true;
```

[2] https://sourceforge.net/
[3] http://maxima.sourceforge.net/docs/manual/maxima.html

3.2 JavaScript

JavaScript-ohjelmointikieltä voidaan käyttää lisäämään verkkosivuille dynaamisuutta ja vuorovaikutusta käyttäjän kanssa. Vaikka JavaScript ei varsinaisesti ole kaikkeen pystyvä ohjelmointikieli, voidaan sillä kirjoittaa käytännössä kokonaisia ohjelmia, jotka ajetaan verkkoselaimella. (Negrino ja Dori, 2007). Tämän tutkielman loppupuolella esiteltävä JavaSketchpadToJSXGraph-kääntäjä on toteutettu täysin JavaScript-kielellä, jotta kääntäjän käyttäminen selainohjelmilla olisi sujuvaa.

Koska STACK-tehtäviä rakennetaan ja suoritetaan web-selaimella HTML-pohjaisessa oppimisympäristössä kuten Moodle, JavaScript-ohjelmoinnilla tehtäviin on luontevaa ja kätevää luoda dynaamisuutta. Myös vastausten tarkistamistapaa voidaan hyvinkin monipuolisesti muokata JavaScriptiä hyödyntämällä. Tehtävätyypit, joissa opiskelija antaa vastauksensa dynaamista kuviota muuttamalla, ovat myöskin rakennettavissa JavaScript-kieltä käyttäen.

Eräs hyödyllinen ominaisuus, jota JavaScript kielen avulla voidaan hyödyntää on *dokumenttioliomalli (Document Object Model)* eli DOM. Sen avulla web-sivun objekteja voidaan hakea ja muokata sekä niiden arvoja lukea ja muuttaa. (Negrino ja Dori, 2007). Esimerkiksi Moodlen STACK-tehtävään voidaan luoda oma painike, joka piilottaa tai tuo esiin jotain, tai tekee jonkin toiminnon. Listauksessa 1 on esimerkki kuinka DOM toimii: HTML-kielellä luodut painikkeet haetaan käsittelyyn komennolla

```
document.getElementById()
```

ja niiden tyyliominaisuuksia hyödyntäen piilottaa ne.

```
1  <html>
2  ...
3  <input id="IIpr7yes" onclick="ansWrongpr7()" style="
       display:inline-block;" value="Yes" type="button">
4  <input id="IIpr7no" onclick="ansRightpr7()" style="display:inline-block;
       " value="No" type="button">
5  ...
6  </html>
7  <script>
8  ...
9  document.getElementById('IIpr7yes').style.display="none";
10 document.getElementById('IIpr7no').style.display="none";
11 ...
12 </script>
```

Listaus 1: JavaScript-DOM -esimerkki

Grafiikan esittämiseen JavaScript-kielellä käytetään kolmea eri alustaa (sovellusta) verkkoselaimesta riippuen: SVG[4] (Scalable Vector Graphics), VML[5] (Vector Markup Language) ja Canvas[6]. Eri selaimet käyttävät eri esityssovelmaa. Kuvasta 4 näemme, että uudemmat selaimet tukevat hyvin SVG rakennetta.

JavaScript on ainoa mahdollinen ohjelmointikieli, kun halutaan luoda selainpohjaista dynaamista matematiikkaa, joka toimii kaikilla laitteilla (Crockford, 2008). JavaScriptin alkuvaiheissa

[4] https://www.w3.org/TR/SVG/
[5] https://www.w3.org/TR/NOTE-VML
[6] https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial

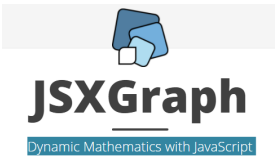
	Firefox 3+	Internet Explorer		Opera	Safari	Chrome	
		4-8	9		incl. iPad	Desktop	Android $\leq$ 2.2
SVG	✓	–	✓	✓	✓	✓	–
VML	–	✓	✓	–	–	–	–
Canvas	✓	–	✓	✓	✓	✓	✓
market share	22.93%	60.4%	–	2.37%	5.16%	7.52%	

Kuva 4: Esitysteknologia eri selaimissa, kuvan lähde (Market Share Reports, 2010).

selaimella suoritettava koodi saattoi toimia hyvin hitaasti, mutta nykyisin JavaScript ohjelmien käynnistysaika on lähellä nollaa (Gerhauser ym., 2011).

3.3 JSXGraph

Internetin yleistyessä 1990-luvun loppupuolella tulivat graafiset web-selaimet tärkeiksi työkaluiksi myös matematiikan opettamisessa ja graafisessa esittämisessä. Oracle-yhtiön[7] kehittämä Java oli tärkeä ja paljon käytetty ohjelmointikieli geometriaa ja laskentaa suorittavien sovellusten ohjelmoinnissa. Java oli aikoinaan hyvä alusta *appleteille*, verkkoselaimilla suoritettaville sovelmille. Nämä sovellukset vaativat kuitenkin Javan virtuaalikoneen ladattavaksi ja ajettavaksi verkkoselaimen taustalle, jolloin Java-applettien toimiminen käytännössä rajoittuu pöytätietokoneille ja kannettaville tehden applettien katselun nykyisin yleistyville älypuhelimille ja tableteille hankalaksi, joissain laitteissa jopa mahdottomaksi. (Gerhauser ym., 2011). Lisäksi monet selainohjelmistot ovat lopettaneet Java-tuen. Myös Oracle on ilmoittanut lopettavansa tuen Java-liitännäisille tulevissa ohjelmistopäivityksissään. (Topic, 2016).



Kuva 5: JSXGraph-logo

JSXGraph[8] on Bayreuthin yliopiston kehittämä JavaScript-ohjelmakirjasto interaktiivisen geometrian, funktioiden piirtämisen, taulukoinnin ja datan visualisoinnin luomiseen kaikilla web-selaimilla. Paitsi että JSXGraph toimii kaikilla verkkoselaimilla, toimii se lähestulkoon kaikilla muillakin alustoilla ja laitteilla, joissa on graafinen web-selain (Gerhauser ym., 2011). JSXGraph on kokonaisuudessaan JavaScript-kielellä ohjelmoitu verkkoselainten liitännäinen, eikä vaadi tai tukeudu muihin JavaScript kirjastoihin. Ensisijaisesti JSXGraph käyttää SVG-esitystä grafiikan piirtämiseen, ja mikäli selain ei sitä tue, käyttää JSXGraph VML- tai Canvas-grafiikkaympäristöä (Gerhauser ym., 2011). Erityisen hyvä ominaisuus JSXGraphissa on se, että sen käyttö ei vaadi päivitystä tietokoneen laitteistolta eikä tietokoneelle tarvitse asentaa ohjelmistoja (Gerhauser ym., 2011). Lisäksi kuvat latautuvat hyvin nopeasti verrattuna esimerkiksi Java-appletteihin.

JSXGraph-sovelluksen tavoite on olla todella dynaaminen (Gerhauser ym., 2011). JSXGraphilla on helppo luoda objekteja siten että niiden ominaisuudet riippuvat muista objekteista. Elementtien väliset suhteet voidaan halutessa määritellä JavaScript-funktioilla, joiden rakenne voi olla hyvinkin monimuotoinen sisältäen esimerkiksi ehtolauseita. JSXGraph-kuvioita koodatessa tehtävän laatijan ei tarvitse ladata sovellusta tietokoneelleen, vaan riittää kiinnittää JSXGraph-kirjasto HTML-koodiin. JSXGraph kirjasto löytyy mm. sivustolta Sourceforge[9].

[7] https://www.oracle.com/index.html
[8] http://jsxgraph.uni-bayreuth.de/wp/index.html
[9] https://sourceforge.net/projects/jsxgraph/

Lisäksi on hyvä ladata muotoilutiedosto *jsxgraph.css*, joka myös löytyy Sourceforgesta ladattavasta paketista. Tämä muotoilutiedosto sisältää joitakin JSXGraph-kuva-alueiden ulkoasuun vaikuttavia määrittelyjä muttei ole pakollinen, sillä muotoiluasetuksia voi määritellä myös itse. CSS-muotoilua (Cascading Style Sheets) ei käsitellä tässä tutkielmassa, ja hyviä oppaita löytyy internetistä[10]. Listauksessa 2 on yksinkertainen esimerkki JSXGraph-kirjaston ja muotoilun lataamisesta ja kuvion rakentamisesta HTML-tiedostoon koodaamalla.

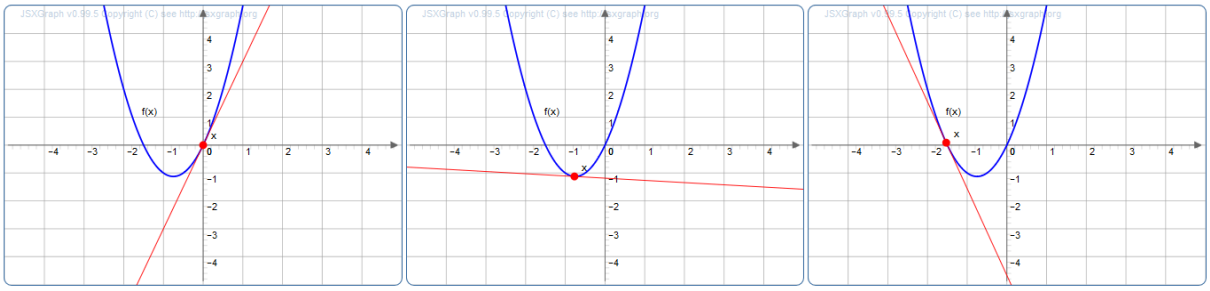
```
1
2  <html>
3  <head>
4  <script src='jsxgraphcore.js'></script>
5  <link rel='stylesheet' type='text/css' href='jsxgraph.css'>
6  <title>JSXGraph esimerkki</title>
7  </head>
8  <body>
9  <div class="jxgbox" id="Taso" style="width: 500px; height: 350px;
       overflow: hidden; position: relative;"> </div>
10 <script>
11 var taulu = JXG.JSXGraph.initBoard('Taso', {boundingbox: [-5,5,5,-5],
       keepaspectratio: false, axis: true, showCopyright: true,
       showNavigation: false, grid: true} );
12 var kuvaaja = taulu.create('functiongraph', [function(x){return 2*
       Math.pow(x,2)+3*x;} ],{name:'f(x)', withLabel:true, strokeWidth:2} )
       ;
13 var piste = taulu.create('glider', [0, 0, kuvaaja], {name:'x'});
14 taulu.create('functiongraph', [function(x){return (4*piste.X()+3)*(
       x-piste.X()) + 2*Math.pow(piste.X(),2)+3*piste.X();}, ], {color:'red
       '} );
15 </script>
16 </body>
17 </html>
```

Listaus 2: JSXGraph-esimerkkikoodi

Tarkastellaan seuraavaksi JSXGraph-syntaksia käymällä Listauksen 2 koodi läpi oleellisilta osin. Rivi 4 sisältää JSXGraph-ohjelmakirjaston (*Framework*) lataamisen selaimeen, ja tässä kirjaston `jsxgraphcore.js` lähde voi olla myös verkko-osoite. Rivillä 5 ladataan muotoiluasetukset. Rivillä 9 luodaan verkkosivulle lohko (`div`) kuviota varten. Tässä on määriteltävä pikseleinä kuva-alueen koko HTML-sivulla. Tässä voidaan asettaa myös muita muotoiluasetuksia, jos niitä ei ladata (rivillä 5) tai niitä halutaan muuttaa. JSXGraph-kuvion luokka (`class`) on aina `jxgbox` ja tunniste (`id`) on yksilöllinen kuviolle varattavan HTML-lohkon nimi. Lisäksi kuva-alueelle varataan tässä sopiva koko. JSXGraph-käskyt kirjoitetaan JavaScriptin tapaan `script`-tagien väliin. Rivillä 11 alustetaan muuttuja `taulu`, joka luo JSXGraph-taulun (*board*), jonka ulkoasua voidaan muokata useilla optioilla, esimerkiksi näytetäänkö akselit. Taulun lohko (rivillä 9) HTML-koodissa löydetään tunnisteen avulla.

JSXGraph-objektit voidaan halutessa asettaa muuttujiksi. Tässä `taulu`, `kuvaaja` ja `piste` alustetaan muuttujiksi, joihin voidaan myöhemmin viitata. JavaScript-kielessä muuttujien tietotyyppejä ei erotella, vaan muuttujat alustetaan yleisesti tyypiksi *var*. Rivillä 12 muuttujaksi `kuvaaja` luodaan objekti tyyppiä *functiongraph*, jota voidaan yleisesti käyttää funktioiden kuvaajan piirtämiseen. Rivillä 13 luodaan `piste`, joka on objekti tyyppiä *glider* ja on toiminnoltaan liukuva piste kiinnitettynä haluttuun objektiin.

[10]Kattava CSS-opas: `http://www.csstutorial.net/`



Kuva 6: JSXGraph-esimerkki

Kuvassa 6 on Listauksen 2 koodi tarkasteltuna web-selaimella kolmessa eri tilanteessa, dynaamisesti kuvan objekteja muutettuna. Nyt pistettä siirtämällä JSXGraph luo rivin 15 funktion kuvaajan, joka on tässä tapauksessa paraabelin tangentti kyseisessä pisteessä. Tästä näemme JSXGraph-ohjelman toiminnallisuuden lisäksi dynaamisen geometrian luonteen yleisesti: pistettä voidaan liikutella vain muuttujan `kuvaaja` muodostamalla polulla, ja tangentin kuvaajan muodostava komento on yhteydessä pisteen sijaintiin kuviossa. Periaate on sama kaikissa dynaamisen geometrian sovelluksissa.

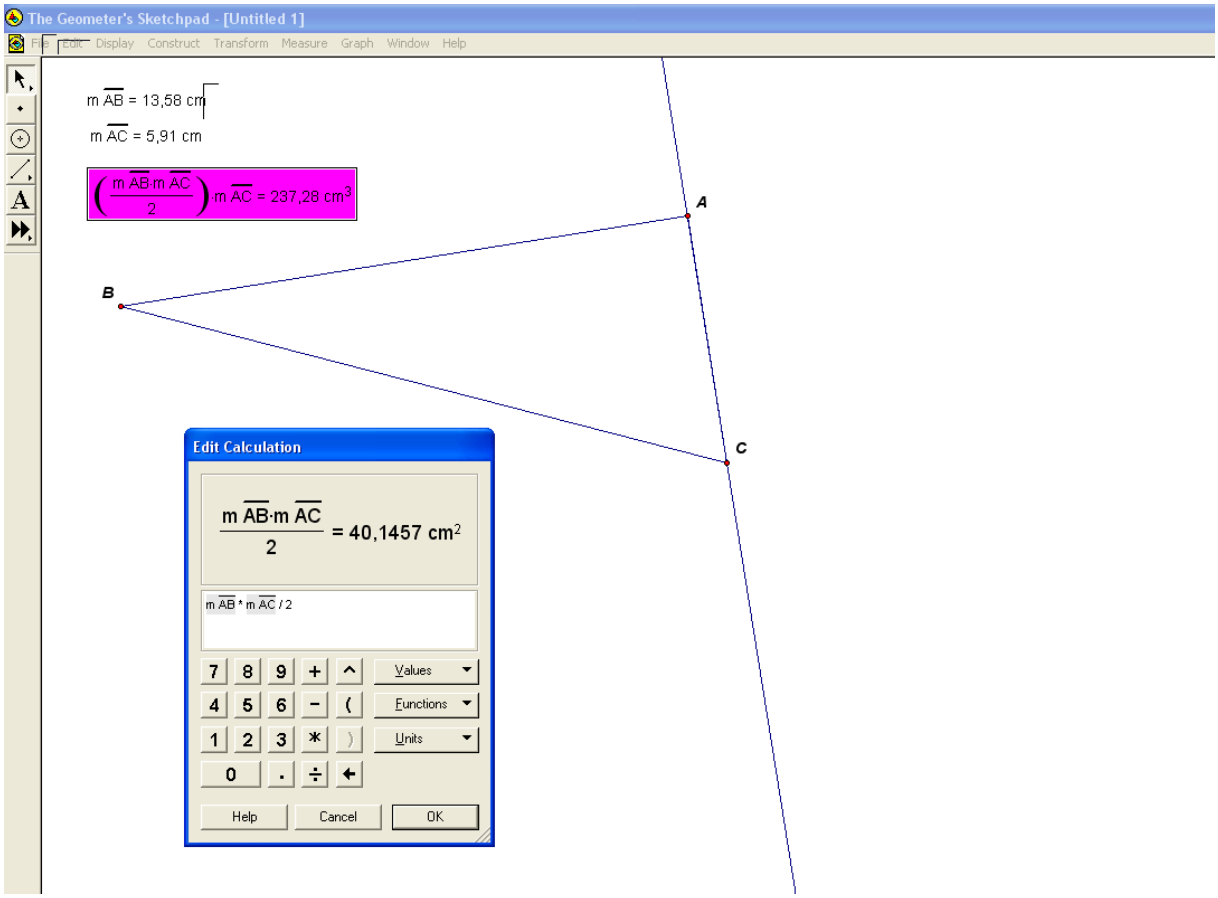
Koska tässä tutkimuksessa konstruoidut STACK-tehtävät ovat rakenteiltaan web-pohjaisia, mahdollistaa JSXGraph graafisten matemaattisten kuvioiden upottamisen kysymyksiin, ja sitä myöten dynaamisten ja visuaalisten tehtävätyyppien laatimisen. Merkittävä etu on myös se, että STACK-kysymyksissä Maximalla määriteltyjen muuttujien arvoja voidaan käyttää JSXGraph-koodissa objektien konstruktioissa vaikkapa pisteen koordinaatteina tai ympyrän säteenä. Lisäksi JSXgraph-koodissa voidaan kuvioon rakennetuille elementeille asettaa *tapahtumankäsittelijä (event handler)*, jonka avulla käyttäjän toimintaa voidaan seurata, ja siten suorittaa määrättyjä toimintoja (funktioita). Tämä mahdollistaa sen, että STACK-järjestelmän tietokoneavusteinen tarkastus voidaan yhdistää JavaScript-koodilla JSXGraph-kuvioihin, jolloin opiskelijan syöte voidaan tarkistaa oikeaksi tai vääräksi graafisesti annettujen ehtojen mukaan. Tähän palataan Luvussa 5.4.

3.4 The Geometer's Sketchpad ja JavaSketchpad

The Geometer's Sketchpad[11] on Key Curriculum Press-yhtiön julkaisema dynaamisen geometrian työkalu. Ohjelmaa on kehitetty 1980-luvulta saakka visuaalisen geometrian projektissa Swarthmoren yliopistossa tohtoreiden Eugene Klotz ja Doris Schattschneider johdolla (The Geometer's Sketchpad Resource Center, 2014) (Bindner ja Martin, 2011). Projektin tarkoitus oli kehittää uutta materiaalia geometrian opetukseen, ja projektin suojissa Nicholas Jackiw rakensi ensimmäisen version ohjelmistosta (The Geometer's Sketchpad Resource Center, 2014). Se on vakiinnuttanut asemansa matematiikan opetuksessa erityisesti Amerikassa.

The Geometer's Sketchpad -sovellus sisältää graafisen käyttöliittymän, jonka avulla voidaan luoda interaktiivisia kuvioita (sketch). Sovellus sisältää työkaluja pisteiden, janojen, ympyröiden, ym. elementtien rakentamiseen. Lisäksi sovelluksella on mahdollista luoda laskennallisia arvoja, jotka perustuvat elementtien ominaisuuksiin kuviossa. Esimerkiksi janan pituus voidaan näyttää ruudulla lukuarvona, joka päivittyy välittömästi, jos janan pituutta muutetaan.

[11] `http://www.keycurriculum.com/sketchpad.1.html`



Kuva 7: Geometer's Sketchpad -sovellusikkuna, johon luotu Listauksen 5 koodia vastaava konstruktio

JavaSketchpad[12] (JSP) on sovellus, jolla The Geometer's Sketchpadilla luotuja kuvioita voidaan julkaista internetissä. Näin ollen kuka tahansa kenellä on Java-tuettu internetselain voi tarkastella ja vuorovaikuttaa kuvion elementtien kanssa ilman, että tarvitsee omistaa The Geometer's Sketchpad sovellusta (Bindner ja Martin, 2011). JSP on tarkemmin sanottuna Java-appletti, jolla The Geometer's Sketchpad -kuvioita voidaan näyttää verkkoselaimessa. JSP-kuviot koodataan HTML-tiedostoon sen omalla syntaksilla. Syntaksista ja koodin kirjoittamisesta HTML-sivulle on esimerkki Listauksessa 3. Koodia voi kirjoittaa HTML-dokumenttiin myös käsin, jolloin JSP-kuvioita voidaan rakentaa ilman, että The Geometer's Sketchpad -sovellusta tarvitaan lainkaan.

JSP-syntaksissa kuvion elementit koodataan riveittäin. Jokainen objekti asetetaan omalle rivinumerolleen ja objekteihin voidaan viitata niiden rivinumeroilla. Listauksen 3 koodissa jana (`Segment`) rivillä `{3}` konstruoidaan viittaamalla pisteisiin riveillä `{1}` ja `{2}`. Koodia luetaan ylhäältä alaspäin ja vain aiemmin määriteltyihin objekteihin voidaan viitata.

JSP-koodissa voidaan määritellä *laskennallisia arvoja* sisältäviä elementtejä kuvioon. Laskennalliset arvot ovat kuvion elementtien ominaisuuksista laskemalla muodostettuja lukuarvoja. Esimerkiksi JSP-konstruktio `Ratio/Points` tuottaa laskennallisen arvon kolmen pisteen etäisyyksien suhteen siten, että jos pisteet ovat A, B ja C, niiden etäisyyksien suhde saadaan laskemalla $|AC|/|AB|$. Toinen esimerkkikonstruktio on `Calculate`, jonka avulla käyttäjä voi itse muodostaa monenlaisia laskulausekkeita, joissa voidaan käyttää vakioita ja muiden las-

[12] `http://www.keycurriculum.com/node/704.html`

kennallisten arvojen lukuarvoja muuttujina. Vakioiden, laskennallisten arvojen ja aritmetiikan lisäksi `Calculate`-operaatiossa voidaan käyttää standardeja alkeisfunktioita, ja lausekkeessa käytetään käänteistä puolalaista merkintätapaa (reverse Polish notation RPN), jossa vältetään sulkeiden käyttö. Laskennallisia arvoja voidaan näyttää käyttäjälle kuviossa, ja niitä voidaan yhdistellä toisiinsa ja muihin objekteihin.

Sebastian Lisken on tehnyt JSP koodista muunnetun rakennuskielen *jsp.awk*, joka helpottaa kuvioiden laatimista suoraan HTML-koodiin kirjoittamalla. Siinä rivinumerot voidaan korvata muuttujanimillä, joiden eteen laitetaan merkki `$`, esimerkiksi `$piste1`. Kattavat ohjeet *jsp.awk*-muotoisten ja JSP-muotoisten kuvioiden rakentamiseen, syntaksiin ja rakenne-elementteihin löytyy Liskenin kokoamalta ohjesivustolta[13].

```
1  <APPLET CODEBASE="jsp" ARCHIVE="jsp4.jar" CODE="GSP.class" WIDTH=400
       HEIGHT=200 ALIGN=Center TextFont="Helvetica" TextBold=1 TextSize=18>
2  <PARAM NAME=Offscreen VALUE=1>
3  <PARAM NAME=Frame VALUE=1>
4  <PARAM NAME=LabelFont VALUE="Courier">
5  <PARAM NAME=LabelBold VALUE=1>
6  <PARAM NAME=MeasureFont VALUE="Courier">
7  <PARAM NAME=MeasureSize VALUE=14>
8  <PARAM NAME=MeasureBold VALUE=1>
9  <PARAM NAME=MeasureInDegrees VALUE=1>
10 <PARAM NAME=DirectedAngles VALUE=0>
11 <PARAM NAME=BackRed VALUE=230>
12 <PARAM NAME=BackGreen VALUE=230>
13 <PARAM NAME=BackBlue VALUE=230>
14 <PARAM NAME=Construction VALUE="
15 {1} FixedPoint(100, 40)[red, label('A')];
16 {2} FixedPoint(100,160)[red, label('B')];;
17 {3} Segment(1, 2)[magenta, thick];
18 ">
```

Listaus 3: JavaSketchpad-syntaksi

3.5 JavaSketchpadToJSXGraph-kääntäjä

3.5.1 Alkuvaiheet

Vuodesta 2002 lähtien laitoksella on tuotettu runsaasti oppimateriaalia JSP-kuvioina, joita on käytetty etenkin kursseilla Matematiikan johdantokurssi, Lineaarialgebra ja Diskreetti matematiikka, sekä Erasmus-opettajavaihdossa. JSP-kuviot vaativat toimiakseen Java-tuetun internetselaimen,ja selainten tuki on heikentynyt tältä osin huomattavasti tietoturvasyistä. Kuviot ovat osoittautuneet tarkoituksenmukaisissa pedagogisissa puitteissa hyviksi apuvälineiksi matemaattisten käsitteiden opetuksessa ja oppimisessa (Pesonen ym., 2006). Ajatus kääntäjästä heräsi lounaspöydässä ohjaajani Martti Pesosen kanssa käydyn keskustelun tuloksena; kuinka saisimme tämän suuren määrän JSP-kuvioita muunnettua nopeasti ja rakenteeltaan identtisesti selainten paremmin tukemaan JSXGraph-muotoon. Toki kuvioita voitaisiin myös samalla modifioida ja uusiakin muodostaa.

[13] `https://www.math.uni-bielefeld.de/~lisken/jsp/`

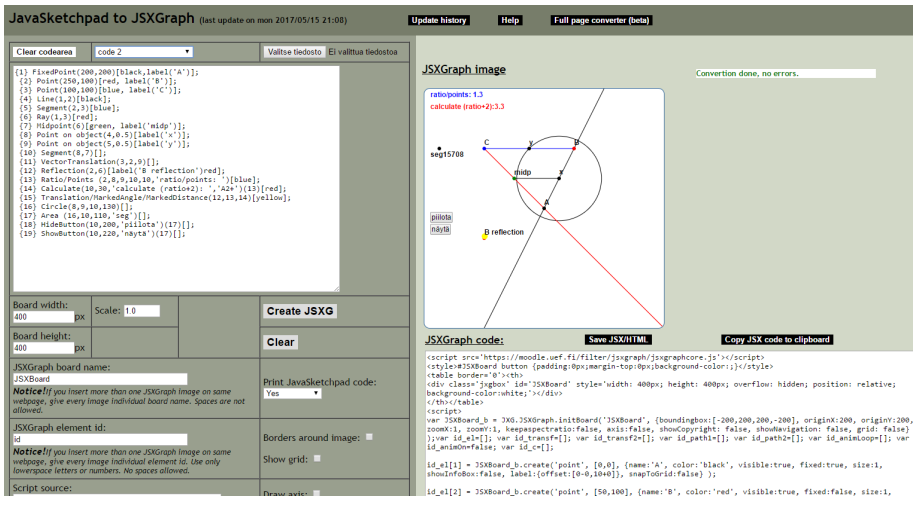
Tämän keskustelun pohjalta lähdin tutkimaan mahdollisuutta toteuttaa kääntäjä. Koska käytettyjen DGE-sovellusten syntaksit muistuttavat toisiaan, kääntäjän rakentaminen olisi melko suoraviivaista. JSP-koodi puretaan osiin riveittäin, rivien rakenne jaetaan osiin, ja nämä erotellut konstruktiot rakennetaan uudelleen JSXGraph syntaksilla. Lisäksi täytyi pohtia millä kielellä käännössovellus olisi paras rakentaa. JavaScript osoittautui hyväksi ratkaisuksi etenkin seuraavista syistä:

- sovellus toimii palvelimella HTML-dokumenttina, jolloin kääntäjän uusien versioiden jakamisesta käyttäjille ei tarvitse huolehtia
- sovellusympäristö on web-selain, jolloin käännettyä kuvaa voidaan samantien tarkastella selainympäristössä
- mahdollisuus oppia kieli, koska JavaScript ei ollut entuudestaan tuttu, ja sen osaaminen on varsin hyödyllistä

JSPtoJSXGraph-kääntäjä löytyy osoitteesta
`http://cs.uef.fi/matematiikka/ABACUS/JavasketcpadToJSXgraph/`.

3.5.2 Kääntäjän rakenne

Alkuperäisen JSP-koodin rivit erotellaan tietorakenteisiin, jotka sisältävät kunkin elementin attribuutit ja tiedot. Tämän tietojen erottelun jälkeen tiedoista kootaan JSXGraph-syntaksin mukainen koodi, joka voidaan sellaisenaan kopioida suoraan HTML-tiedostoon tai vaikkapa Moodle-kysymykseen. Kääntäjä luo koodista saman käännösprosessin tuloksena sekä käännetyn JSXGraph-koodin että mallin lopullisesta kuvasta. Näin ollen kääntäjää voidaan käyttää myös itsenäisenä ohjelmana, jolla voidaan rakentaa JSXGraph-kuvioita käyttäen kielenä JSP-syntaksia.



Kuva 8: JavaSketchpad-koodi vasemmalla laatikossa, kääntäjän rakentama mallikuvio sekä JSXGraph-koodi oikealla.

JavaSketchpad- ja JSXGraph-syntaksit ovat samankaltaisia. Molemmat voi kuvata seuraavassa muodossa:

```
Objektin tyyppi [obj Argumentit] [obj ominaisuudet]      (1)
```

Otetaan esimerkiksi JSP-syntaksin mukainen komento

```
{1} Point(200,100)[ black, label('0')];.
```
(2)

Osiin purettu koodi muodostuu objektin ominaisuuksista, joita ovat rivinumero `1`, konstruktiopiste `Point`, paikka kuvassa `(200,100)`, väri `black` ja pisteen `label` eli nimi `'0'`. Kääntäjä luo jokaisen puretun rivin osista olion, joka sisältää attribuutteinaan JSP-koodin riviltä luetut tiedot, jotka se sitten vaihtaa JSXGraph-muotoon. Kaikki oliot tallennetaan kääntäjässä taulukoksi, jonka jokaista indeksiä vastaa alkuperäisen JSP-koodin rivinumero. Näin JSP-koodissa tehdyt viittaukset aikaisemmin määriteltyihin konstruktioihin toimivat selkeästi myös kääntäjällä luodussa koodissa. Lopullinen käyttäjälle näkyvä JSXGraph-koodi tuotetaan taulukon olioiden attribuuteista. Tarkastellaan Listauksen 3 yksinkertaista JSP-koodia, jossa on määritelty elementit

```
{1} Point(200,100)[ black, label('A')];
{2} Point(100,100)[ black, label('B')];
{3} Segment(1,2)[ black, thick];
```

Kääntäjä luo elementeistä taulukon, jonka kukin olio sisältää rivinumeroltaan vastaavien rivien tiedot. Kääntäjällä luotu Listausta 3 vastaava JSXGraph-koodi on Listauksen 4 mukainen.

```
id_el[1] = JSXBoard_b.create('point', [0,100], {name:'A', color:'black',
   visible:true, fixed:false, size:1, showInfoBox:false, label:{offset:
   [0-0,10+0]}, snapToGrid:false} );
id_el[2] = JSXBoard_b.create('point', [-100,100], {name:'B', color:'black',
    visible:true, fixed:false, size:1, showInfoBox:false, label:{offset:
   [0-0,10+0]}, snapToGrid:false} );
id_el[3] = JSXBoard_b.create('segment', [id_el[1],id_el[2]], {name:'',
   color:'black', visible:true, strokeWidth:3, showInfoBox:false, label:{
   offset:[0-0,10+0]}, lastArrow:false, firstArrow:false} );
```

Listaus 4: JSXGraph-esimerkki, Listauksen 3 käännetty koodi

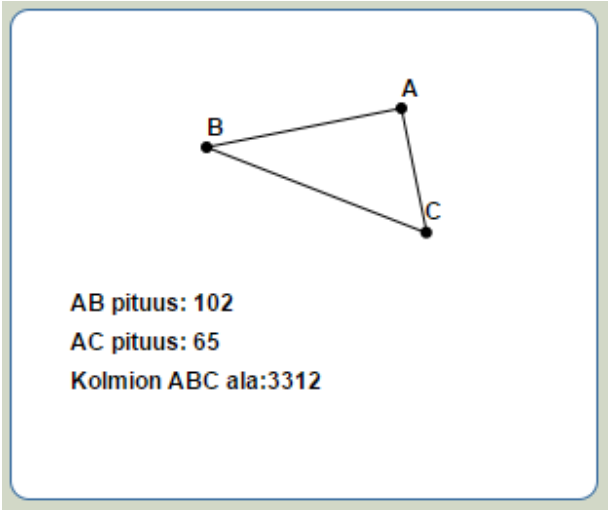
Luvussa 3.4 todettiin, että JSP-kuvioissa voidaan käyttää ja yhdistellä kuvion objektien ominaisuuksista muodostettuja laskennallisia arvoja. JSXGraph-syntaksissa kaikille näille ei ole vastiketta, joten tällaisten JSP-koodin rivien kääntäminen vaatii erillisen algoritmin. Tarkastellaan Listauksen 5 JSP-koodia:

```
{1} Point(200,50) [black, label('A')];
{2} Point(100,70) [black, label('B')];
{3} Segment(1,2)[ black];
{4} Perpendicular (1,3) [black, hidden];
{5} Point on object (4,1) [black, label('C')];
{6} Segment (2,5) [black];
{7} Segment (1,5) [black];
{8} Length (3,30,150,'AB pituus: ') [black];
{9} Length (7,30,170,'AC pituus: ') [black];
{10} Calculate(30,190,'Kolmion ABC ala: ', 'AB*2/')(8,9)[];
```

Listaus 5: JSP-esimerkki, laskennalliset konstruktorit

Se tuottaa kuvion (vrt. Kuva 7), jossa on suorakulmainen kolmio ja kolme laskennallista arvoa; Length riveillä 8 ja 9, sekä Calculate rivillä 10. Calculatessa viitataan rivien 8 ja 9 laskettuihin janojen pituuksiin.

Nämä arvot ovat dynaamisia siinä mielessä, että niiden arvo muuttuu samalla kun janoja muutetaan kuviossa. JSXGraph-syntaksissa ei ole suoria vastikkeita kaikille näille komennoille. Kääntäjässä tämä on ratkaistu siten, että jokaisesta laskennallisesta arvosta muodostetaan funktio, joka tunnistetaan kuten muutkin objektit rivinumeron perusteella. Funktion lisäksi muodostetaan toinen rivi, joka on tyypiltään text-muotoinen JSXGraph-objekti, ja jonka avulla laskettu tulos voidaan näyttää. Listauksessa 6 on Listauksesta 5 käännetty koodi rakennettujen objektien osalta.



Kuva 9: Listauksesta 5 käännetty kuvio.

```
1  id_1_el[1] = JSXBoard_b.create('point', [0,150], {name:'A',label:{
       offset:[0,10]},color:'black',size:1,showInfoBox:false});
2  id_1_el[2] = JSXBoard_b.create('point', [-100,130], {name:'B',label:{
       offset:[0,10]},color:'black',size:1,showInfoBox:false});
3  id_1_el[3] = JSXBoard_b.create('segment', [id_1_el[1],id_1_el[2]],{
       strokeColor:'black',strokeWidth:1});
4  id_1_el[4] = JSXBoard_b.create('perpendicular', [id_1_el[1],id_1_el
       [3]],{strokeColor:'black',strokeWidth:1,visible:false});
5  id_1_el[5] = JSXBoard_b.create('glider',
       [30.238386116430217,4.80806941784893,id_1_el[4]], {name:'C',label:{
       offset:[0,10]},color:'black',size:1,showInfoBox:false});
6  id_1_el[6] = JSXBoard_b.create('segment', [id_1_el[2],id_1_el[5]],{
       strokeColor:'black',strokeWidth:1});
7  id_1_el[7] = JSXBoard_b.create('segment', [id_1_el[1],id_1_el[5]],{
       strokeColor:'black',strokeWidth:1});
8  function id_18() {return id_1_el[3].L();} id_1_el[8] = JSXBoard_b.create
       ('text', [-170,50,function(){return 'AB pituus: ' + Math.round(id_18
       ()*10)/10;}], {name:'', color:'black'} );
9  function id_19() {return id_1_el[7].L();} id_1_el[9] = JSXBoard_b.create
       ('text', [-170,30,function(){return 'AC pituus: ' + Math.round(id_19
       ()*10)/10;}], {name:'', color:'black'} );
10 id_1_el[10] = JSXBoard_b.create('text', [-170,10,function(){return '
       Kolmion ABC ala:' + Math.round((((id_18()*id_19())/2))*10)/10 + ''
       ;}],{name:'', color:'black', fixed:true});
```

Listaus 6: Listauksen 5 käännetty koodi

Riveillä 8 ja 9 on JSP-komentoa `Length` vastaavat käännetyt JSXGraph-koodit. Ensin luodaan funktio, joka laskee ja palauttaa objektin tyyppiä vastaavan arvon. Lisäksi asetetaan JSXGraph-

tekstiobjekti, jossa tätä funktiota kutsutaan näytettäväksi tekstiksi kuvioon. Rivillä 10 on operaatiota `Calculate` vastaava kääntäjän muodostama koodi, johon kääntäjä on muodostanut JSP-koodin komentoa vastaavan laskutoimituksen. Funktioiden `id_18` ja `id_19` palauttamaa arvoja käytetään laskemaan rivin 10 tekstielementin arvo.

Useimmille JSP-operaatioille löytyy vastine JSXGraph-kirjastosta, mutta eräiden rakenteeltaan eriävien operaatioiden kohdalla kääntäjään on pyritty luomaan JSP-komentoa vastaava ratkaisu. Niihin palataan Luvussa 3.5.4. Kääntäjän rakennetta voi tarkastella sen dokumentaatiosta[14].

3.5.3 Kääntäjän käyttö

Käännettävä JSP-koodi kirjoitetaan tai kopioidaan käyttöliittymän vasemmassa yläkulmassa olevaan tekstikenttään. Käyttöliittymän vasemmasta reunasta löytyy runsain määrin asetuksia, joilla kuviota voidaan säätää. Oleellisimmat asetettavat tiedot ovat laadittavan kuvion leveys ja korkeus pikseleinä. Kääntäjä osaa melko hyvin hakea nämä kuvan kokoon liittyvät tiedot myös suoraan JSP-koodista mikäli ne siihen sisältyvät. Myöskin JSXGraph-koodin elementtien nimiä on mahdollista yksilöidä asetusten avulla. Tämä on välttämätöntä mikäli halutaan esittää useampi JSXGraph-kuvio samalla HTML-sivulla, jotteivat muuttujien nimet eri kuvioissa ole samoja ja näin ollen aiheuta ristiriitoja.

Kääntäjä tuottaa koodin ja mallikuvion käyttöliittymän oikeaan reunaan. Käännetty koodi voidaan kopioida ja liittää haluttuun HTML-dokumenttiin tai Moodle-kysymykseen. Kääntäjässä on valittavana elementeille myös joitakin JSXGraph-ohjelmakirjaston tarjoamia lisäoptioita, joita ei JSP-koodissa ole tarjolla. JSPtoJSXGraph-kääntäjän käyttöön ja lisäoptioihin voi tutustua tarkemmin kääntäjän ohjesivustolla[15].

3.5.4 Haasteet ja rajoitukset

Joillekin JavaSketchpad-objektityypeille ei ole suoraa vastinetta JSXGraph-ohjelmakirjastossa. Esimerkiksi pisteen animointi jollakin polulla on toteutettu kääntäjässä erikseen rakennetulla algoritmilla. JSXGraph ei sisällä operaatiota, jolla muista objekteista riippumaton piste saataisiin liikkumaan animoidusti jotain määrättyä polkua (esimerkiksi ympyrän kehää) pitkin. Kääntäjä muodostaa käännettyyn koodiin erillisen funktion, jonka avulla piste lukitaan polulle (kohde objektiin), lasketaan pisteet, joihin sitä siirretään (animoidaan) ja aloitetaan animointi. Funktion uudelleenkutsuminen irrottaa pisteen objektista, johon se väliaikaisesti kiinnitettiin ja pysäyttää animoinnin.

Toinen samankaltainen JSP-operaatio on `Locus`, jolle ei ole suoraa vastinetta JSXGraph-ohjelmassa. JSP-operaatio `Locus` laskee ja piirtää pisteen P_2 uran, joka muodostuu pisteen P_1 liikkuessa suoralla tai ympyrän kehällä, kun piste P_2 on pisteen P_1 translaatiopiste. JSXGraph-operaatiolla `tracecurve` voidaan piirtää polku vastaavasti, mutta vain jos piste P_1 on määritelty siirreltäväksi ainoastaan suoralla tai ympyrän kehällä (Point on object). JSXGraph-koodiin luodaan käännösoperaatiossa algoritmi, joka toimii vastiineena JSP-syntaksin `Locus`-komennolle. Listauksessa 7 on hahmoteltu pseudokoodilla algoritmin toimintaperiaate.

[14] `http://cs.uef.fi/matematiikka/ABACUS/JavasketcpadToJSXgraph/doc/index.html`

[15] `http://cs.uef.fi/matematiikka/ABACUS/JavasketcpadToJSXgraph/help.html`

```
locus = [tyhjä lista];
funktio muodostaLocus() {
  locus.remove(); //poistaa (mahdollisen) aiemman locuksen
  asetetaan piste P1 kiinni määritetylle polulle;
  ticks = (1 / n); /* n = locuksen pisteiden määrä */
  P2_koord = [tyhjä lista];
  i=0;
  Kun ticks <= 1 {
    P2_koord[i]= pisteen P2 koordinaatit [P2.X, P2.Y];
    siirretään pistettä P1 polulla jolle se asetettiin ticks verran;
    ticks = ticks + 1 / Locuksen tarkkuus;
    jos (listan P2_koord koko > 0) locus.lisää(jana pisteestä P_2koord[i-1]
        pisteeseeen P2_kood[i]);
  }
  irroitetaan piste P1 polulta;
}
kun pistettä P1 tai sen polkua siirretään { muodostaLocus(); }
```

Listaus 7: JSP-komennon `Locus` toimintaperiaate JavaSketchpadToJSXGraph-kääntäjässä

Locus siis muodostetaan piirtämällä lyhyitä janoja pisteen P_2 uralle riittävän tiheään asetettujen pisteiden välille. Tämä keinotekoisesti muodostettu locus päivitetään sitten aina, kun pistettä P_1 tai sen uraa siirretään.

Eräs rajoitus kääntäjässä on JavaSketchpad-kuvioiden ominaisuus, jolla elementit voidaan asettaa määrätyllä tavalla päällekkäin. Elementeille voidaan asettaa optio `Layer(`*`arvo`*`)`, jossa *`arvo`* määrittelee mihin kerrokseen elementti asetetaan. Tämän ominaisuuden avulla voidaan esimerkiksi jokin piste asettaa olemaan kerroksissa ylempänä muihin verrattuna. Tällöin siihen voidaan tarttua hiirellä ensin suhteessa alemmaksi määritettyihin elementteihin, mikäli elementit ovat kuviossa päällekkäin. Ylemmäksi asetettu elementti myös näkyy kuviossa ensisijaisesti, peittäen alleen alemmalle kerrokselle määritellyt objektit. JSXGraph-kuvioissa ei ole mahdollista asettaa yksittäisten objektien kerrosasetusta, ainoastaan kokonaisten objektityyppien kerrosasetus on muutettavissa. Esimerkiksi pisteet voidaan määritellä ylemmäksi kuin suorat. Kääntäjän viimeisin versio[16] ei siis ota JSP-koodin `Layer`-optioita huomioon lainkaan, mutta niitä ei tarvitse käännettävästä koodista kuitenkaan poistaa.

Toistaiseksi JSP-translaatiot toimivat kääntäjässä rajoitetusti vain pisteille. Ainoastaan translaation `VectorTranslation` voi muodostaa myös janalle ja suoralle. Muiden objektien translaatio on kuitenkin kehitteillä. Lisäksi kääntäjässä on rajoitus `Calculate`-komennolle. `Calculate`-komennolla voidaan suorittaa laskutoimituksia, jotka voivat sisältää kuviosta muodostettuja muuttujanarvoja. Esimerkiksi komento

```
Calculate(10, 10, 'Sum is ','AB+C+') (5, 6, 7);
```

suorittaa laskutoimituksen, jossa lasketaan muuttujien `A`, `B` ja `C` arvot yhteen. Muuttujan arvot saadaan vastaavasti elementeistä, joihin viitataan komennon lopussa olevilla rivinumeroilla. Tässä tapauksessa muuttujan `A` arvoksi luetaan tieto riviltä 5, `B` saa arvon riviltä 6 ja `C` riviltä 7. Kääntäjän viimeisimmässä versiossa on mahdollista käyttää vain kuutta muuttujaa laskutoimituksissa, toisin sanoen aakkosia A-F. Myös tähän on kehitteillä parannus.

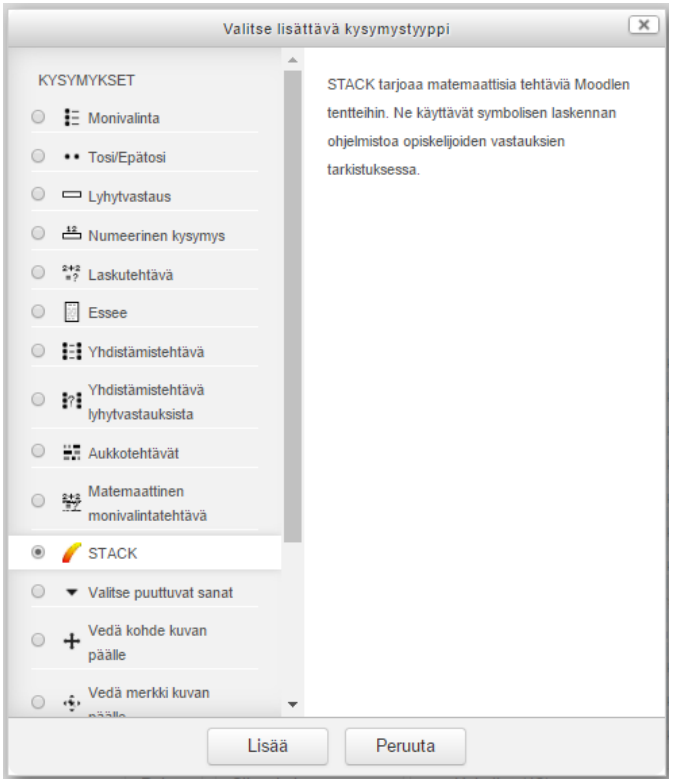
[16] Viimeisin versio päivitetty 5.6.2017

Kääntäjä osaa myös kääntää `$`-muotoisen JSP-koodin ja muodostaa siitä JSXGraph-koodin. Samalla kääntäjä tuottaa perinteisen rivinumeromuotoisen JSP-koodin, joten sitä voi käyttää myös `$`-JSP-koodin muuntajana. Täysin luotettavasti muunnos ei kuitenkaan viimeisimmässä versiossa toimi, sillä esimerkiksi kommenttien muuntamisessa on havaittu ongelmia.

3.6 Moodle

Moodle on useiden yliopistojen käyttämä *vituaalinen oppimisympäristö* (VLE). Moodle tarjoaa erilaisia työkaluja riippuen käyttäjän oikeuksista. Moodlen perusrakenne koostuu *kursseista*, joissa opettajat voivat tarjota opiskelijoille oppimateriaalia ja erilaisia tehtäviä. Kurssien sisältö voi koostua kokonaisesta vuosikurssista, yksittäisestä istunnosta tai jostain siltä väliltä. Kurssin sisältöä voi ohjata yksi tai useampi opettaja. Opiskelijoiden ja opettajien näkymä Moodlesta on käyttöliittymältään lähes identtinen, mutta opettajilla on luonnollisesti monipuolisemmat työkalut oppimateriaalien luomiseen. (MoodleDocs, 2016).

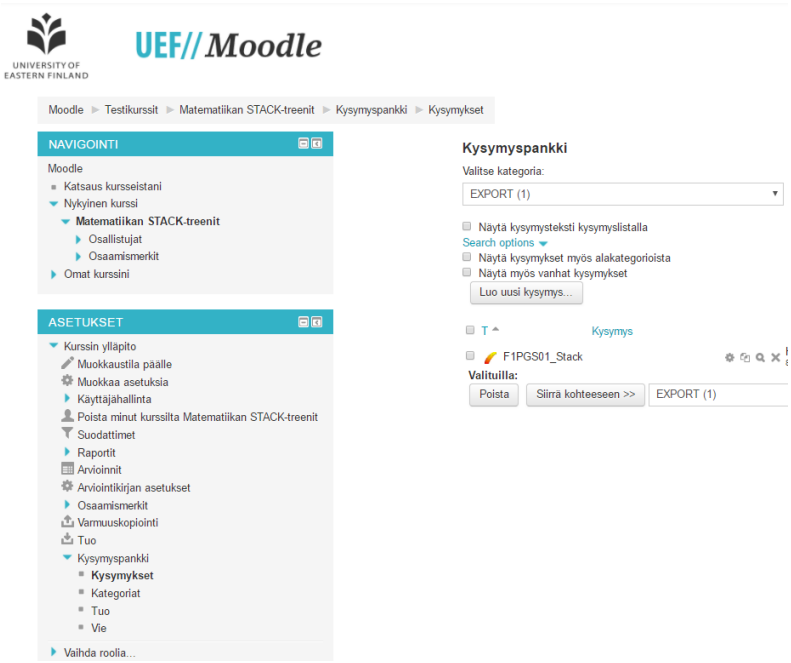
Opettajana Moodleen voidaan tehdä monenlaisia oppimista tukevia aktiviteetteja ja aineistoja, esimerkiksi luoda tietokenttiä, ladata tiedostoja tai vaikkapa keskustelualueita. Eräs aktiviteetti joka voidaan luoda on *tentti* (*quiz*). Tentin tuloksia voidaan tarkastella hyvin yksityiskohtaisesti opiskelijoittain ja kysymyksittäin. Tenttiä voidaan useilla optioilla säätää halutunlaiseksi, kuten asettaa salasana tai tentin sulkeutumisaika. Tentti koostuu yhdestä tai useammasta *kysymyksestä* (*question*), joita voidaan luoda monen tyyppisiä (Kuva 10). STACK on Moodlessa oma kysymystyyppinsä.



Kuva 10: Kysymystyypin valitseminen STACK-kysymykseksi

Moodlessa kysymykset sijoitetaan kurssin *kysymyspankkiin kategorioittain*, ja samassa kategoriassa voi olla kaiken tyyppisiä kysymyksiä (Kuva 11). Kysymyksen laadinta tapahtuu tekstie-

ditoreilla, joita Itä-Suomen yliopiston Moodlessa on valittavana pelkän tekstikenttä-tyyppisen editorin lisäksi kaksi HTML-editoria, Atto sekä TinyMCE[17].



Kuva 11: Kurssin Matematiikan STACK-treenit kysymyspankki ja kategoriat

Vakain selain Moodlen käyttöön on Mozilla Firefox[18]. Moodle on optimoitu ja testattu sillä toimivaksi. Vähiten vakaita selaimia Moodlen käyttöön ovat Safari[19] ja Microsoft Edge[20]. (MoodleDocs, 2016).

Tässä tutkielmassa käytetään usein termejä *kysymys* ja *tehtävä*. On syytä huomata, että näiden termien merkitys eroaa tässä kontekstissa selvästi toisistaan. Puhuttaessa kysymyksestä, tarkoitetaan tässä Moodle-ympäristöön laadittua kysymystä. Tehtävällä sen sijaan tarkoitetaan yleisempää ja usein monimuotoisempaa matemaattista kokonaisuutta kuin mitä Moodle-kysymyksillä voidaan tehdä. Toisaalta Moodlessa on joitakin kysymystyyppejä nimetty käyttäen sanaa tehtävä, esimerkiksi ’Laskutehtävä’ ja ’Yhdistämistehtävä’. Moodle-ympäristöön laadittavista kysymyksistä, olivatpa ne sitten tyyppiä STACK, Laskutehtävä, tai Yhdistämistehtävä, käytetään tässä yhteydessä kuitenkin aina termiä kysymys.

[17] https://www.tinymce.com/
[18] https://www.mozilla.org/fi/firefox/new/
[19] https://safari.en.softonic.com/
[20] https://www.microsoft.com/fi-fi/windows/microsoft-edge

4 STACK

STACK (System for Teaching and Assessment using a Computer algebra Kernel) on Chris Sangwinin kehittämä avoimen lähdekoodin järjestelmä, jolla voidaan luoda ja hienostuneesti arvioida satunnaistettuja verkkopohjaisia matemaattisia tehtäviä (Sangwin, 2012). STACK voidaan kiinnittää Moodle oppimisympäristöön erääksi kysymystyypiksi. STACK on ollut käytössä laitoksella vuoden 2016 alusta lähtien osana ABACUS-projektia.

4.1 STACK-järjestelmän yleisiä ominaisuuksia

STACK-järjestelmä on verkkopohjaisten kysymyssivujen laadintaan suunniteltu ohjelmisto. Järjestelmän rakenne on pyritty tekemään tehtävien laatijan näkökulmasta portaittain monimutkaistuvaksi, jotta perustehtävien formuloiminen olisi mutkatonta (Majander, 2010), mutta monimutkaisemmat tehtävät vaativat käyttäjältä syvällisempää osaamista (Sangwin, 2012). Se on suunniteltu alunperin matematiikan opeuskäyttöön, mutta sen käyttö muidenkin tieteenalojen opetuksessa on hyödyllistä (Rasila ja Sangwin, 2016). Automaattisen tarkastusjärjestelmän luotettavuus on ollut lähtökohtana STACK-järjestelmän kehitykselle (Sangwin, 2012). Järjestelmän on siis pystyttävä opiskelijan matemaattisesta syötteestä (joukko, lista, lauseke ym.) tarkistamaan täyttääkö se kysymyksen kriteerit. Matemaattisesti oikeiksi vastauksiksi voidaan usein tulkita useampi kuin yksi ilmaus, mikä asettaa tarkistamiselle haasteen.

Tietokonealgebrajärjestelmä Maxima käsittelee tehtävissä määriteltyjä muuttujia ja opiskelijan syötteitä. Maximalla voidaan suorittaa laskutoimituksia sekä tehtävänantoa varten että vastauksen tarkistamista varten. Järjestelmällä voidaan luotettavasti tarkastaa myös satunnaistettuja tehtäviä, sillä satunnaiset muuttujat erotellaan tehtäväkohtaisesti ja niitä voidaan käsitellä Maximalla opiskelijan vastauksen ominaisuuksien selvittämisessä. (Sangwin, 2012).

Sangwinin (2012) mukaan STACK-järjestelmän kehityksessä on noudatettu seuraavia yhteisöllisyyttä tukevia periaatteita:

Sisällön ja tunnistautumisen hallinta: Moodle hoitaa tämän tehtävän.

Tietokonealgebrajärjestelmä: Tietokonealgebraohjelmistot ovat laajoja kokonaisuuksia sinällään. Maxima on valittu STACK-järjestelmän ytimeksi luotettavuutensa sekä laajan ja avoimen kehitysyhteisönsä vuoksi.

Matemaattisen kysymyksen luonti: Järjestelmällä on pystyttävä tarpeeksi vaivattomasti luomaan käytännöllisiä ja monipuolisia kysymyksiä, kuitenkin järjestelmän oppimiskynnyksen pysyessä tarpeeksi matalalla. LaTeX-muotoilu ja Maxima-syntaksi toimivat järjestelmässä hyvin yhteen, ja tehtäviin on mahdollista upottaa dynaamista geometriaa vaivattomasti.

Matemaattisen tekstin näyttäminen: STACK-järjestelmä ei itsessään sisällä LaTeX-ladontaan tarvittavia komponentteja, vaan se luottaa Moodle-ympäristön käyttämään LaTeX-muotoiluun. Moodle-ympäristössä mutoilun suorittaa JavaScript-liitännäinen nimeltä MathJax[21], jonka avulla LaTeX-ladontaa voidaan näyttää millä tahansa HTML-sivulla.

[21] https://www.mathjax.org/

4.2 CAS ja CASText

Maxima on STACK-järjestelmässä käytetty tietokonealgebrajärjestelmä (CAS). Sen tehtävä on toimia eräänlaisena järjestelmän moottorina, joka suorittaa laskutoimitukset ja vastausten vertaamisen. Suuri osa Maxima-funktioista on käytössä STACK-systeemissä, mikä mahdollistaa matemaattisten ilmaisujen monipuolisen manipuloinnin. (STACK Documentation, CAS, 2016). Vähänkin monimutkaisimpien kysymysten rakentaminen vaatii jonkin verran Maxima-kielen hallintaa.

Maximan avulla kysymyksiin voidaan luoda yhtälöitä, erisuuruuksia, joukkoja, listoja, matriiseja ja lausekkeita. Lähestulkoon kaikkia Maxima-komentoja voidaan käyttää kysymyksen laatimiseen, mutta osa on kielletty. Kielletty on esimerkiksi komento `define()`. Järjestelmässä voidaan kieltää Maxima-funktioiden käyttö vastauksen syötössä, jottei opiskelija pysty ratkaisemaan tehtäviä käyttäen näitä komentoja. STACK-järjestelmässä määritellään muuttujat seuraavasti:

```
muuttujan_nimi : arvo,
```

esimerkiksi `a : 5;` tai `p : x^2 + 1;`. Muuttuja voidaan määritellä myös käyttäen Maximan funktioita, esimerkiksi

```
p : expand(x+3) * (x-4);
```

Maximassa erityiset vakiot on määritelty `%i`, `%e` ja `%pi`. STACK-järjestelmässä on määritelty samat vakiot pelkillä kirjaimilla, eli `i`, `e` ja `pi`.

CASText on tekstiä, jonka CAS mahdollistaa. CASText on yksinkertaistetusti HTML-dokumentin tekstiä, johon LaTeX- ja CAS-komennot voidaan upottaa (STACK Documentation, CAS, 2016). CAS-komentoja tekstille (muuttujille) voidaan suorittaa ennen kuin kysymys näytetään käyttäjälle. STACK-Moodle -järjestelmässä on useita kenttiä, joihin voidaan kirjoittaa CASText-muotoista tekstiä, esimerkiksi tekstikenttä, johon tehtävänanto kirjoitetaan. STACK ei itsessään sisällä ominaisuuksia tuottaa LaTeX-ladontaa tekstiin, vaan sen hoitaa Moodle ja siinä käytetty MathJax-ohjelmakirjasto. LaTeX-ladontaa voidaan sisällyttää tekstiin kirjoittamalla `\( ... \)` kun kirjoitetaan matemaattisia ilmaisuja riville, ja `\[ ... \]` kun kirjoitetaan omalle keskitetylle kaavariville, eli aivan kuten normaalissa LaTeX-kirjoittamisessa.

Määriteltyjä muuttujia voidaan tulostaa CASText-kenttiin kirjoittamalla `{@ p @}`, missä `p` on määritellyn muuttujan nimi. Näin kirjoitettuna muuttujan arvo kirjoitetaan LaTeX-muodossa samalle riville kuin jolle se asetetaan. Sulkumerkinnän `\( {@ p @} \)` käyttö ei ole tällöin tarpeen, mutta ei kiellettyäkään. Sulkumerkinnän sisälle voidaan kirjoittaa muutakin matemaattista tekstiä, esimerkiksi `\( {@ p @} = x^2 \)`. Mikäli muuttujan arvo halutaan kirjoittaa omalle kaavarivilleen CASText-muotoilussa, täytyy käyttää hakasulkumerkintää `\[ {@ p @} \]`. Mikäli muuttujan arvo halutaan tulostaa (tai lukea) ilman LaTeX-muotoilua, käytetään merkintää `{# p #}`. Tämä merkintätapa on pakollinen silloin, kun muuttujan arvoa halutaan käyttää JavaScript-koodin sisäisenä arvona. Kaikkia määriteltyjä CAS-muuttujia voidaan käyttää missä tahansa CASText-kentässä.

4.3 Kysymysten satunnaisuus

Kysymysten satunnaisuudella tarkoitetaan joidenkin kysymyksen ominaisuuksien arpomista jostakin joukosta. Olennaista kysymysten satunnaistamisessa on, että tehtävän luonne ei satunnaistamisen johdosta muutu (Sangwin, 2012). Vaikka tehtävän joitakin muuttujia siis arvottaisiin, niin lähtökohtaisesti tavoitellaan, että tehtävä sisällöllisesti säilyy samankaltaisena. Esimerkiksi on vaikeustasoltaan eri asia pyytää integroimaan lausekkeet $\sin x$, $\sin^2 x$ tai $\sin^7 x$.

STACK-järjestelmässä muuttujan arvon satunnaistaminen tapahtuu funktiolla `rand()`. Huomioitavaa on, että tämä funktio on ohjelmoitu STACK-Maxima -järjestelmään, eikä se vastaa täysin Maximan omaa `random()`-funktiota. STACK-järjestelmä luo satunnaisen arvon tarkasti määritellyllä tavalla, jotta opiskelijan vastatessa kysymykseen voidaan näyttää tulos kysymyksen samasta versiosta. Tästä syystä Maximan omaa `random()`-funktiota tulee välttää.

Satunnainen muuttujan arvo voidaan luoda muutamilla eri tavoilla.

```
r1: rand(5);
```

arpoo muuttujan `r1` arvoksi kokonaisluvun väliltä 0-4. Komennolla

```
r2: rand([-4, -3, -2, -1, 1, 2, 3, 4]);
```
(3)

arvomme muuttujan `r2` arvoksi luvun joukosta $\{-4,-3,-2,-1,1,2,3,4\}$. Usein selkeämpää on määritellä ensin lista

```
list: [-4, -3, -2, -1, 1, 2, 3, 4];
```

jonka jälkeen voimme kirjoittaa

```
r2: rand(list);
```

jolloin muuttuja `r2` tulee määritellyksi kuten komennolla 3. Funktiolla

```
rand_with_prohib(alaraja, yläraja, lista)
```

satunnaista arvoa voidaan rajata siten, että arvo on satunnainen kokonaisluku väliltä [`alaraja`, `yläraja`] mutta ei mikään listan `lista` alkio. Esimerkiksi komento

```
r2: rand_with_prohib(-4, 4, [0]);
```

määrittelee nyt muuttujan `r2` arvon kuten komento 3. Komento voi sisältää myös aiemmin määrittelemiämme muuttujia. Voimme määritellä muuttujan `r3` arvon esimerkiksi seuraavasti:

```
r3: rand_with_prohib(-4, r1, [r1, r2]);
```

Nyt muuttuja `r3` saa arvon joka on kokoisluku väliltä `[-4, r1]`, mutta ei saa samaa arvoa kuin `r1` tai `r2`.

Lukujen satunnaistamisessa kannattaa käyttää harkintaa varsinkin, jos satunnaistamista halutaan rajoittaa joillain ehdoilla. Esimerkkinä mainittakoon tehtävä, jossa tavoitteena on laskea satunnaisesti luodun matriisin determinantti niin, että determinantti on aina nolla. Eräs tapa luoda tällaiset satunnaiset matriisin alkiot, on muodostaa yksi matriisin sarake muiden sarakkeiden lineaarikombinaationa. Esimerkiksi tällaisen satunnaistetun 3×3 -matriisin määrittely voidaan tehdä asettamalla matriisin kaksi saraketta valmiiiksi määritellyksi, ja kolmas määritellään kahden muun lineaarikombinaationa.

```
s1: rand( [[2,4,5], [-2,-3,5], [3,-2,4]] );
s2: rand( [[5,-5,3], [2,-2,-3], [-3,3,5]] );
k1: rand( [-2,-1,2] );
k2: rand_with_prohib(-2,2, [0,1,k1]);
s3: [k1*s1[1] + k2*s2[1], k1*s1[2] + k2*s2[2], k1*s1[3] + k2*s2[3]];
m1: matrix( [s1[1],s2[1],s3[1]], [s1[2],s2[2],s3[2]], [s1[3],s2[3],s3[3]]);
```

Listaus 8: Satunnaistettu 3×3 -matriisi, jonka determinantti on aina nolla

Listauksessa 8 on määritelty kaksi matriisin saraketta `s1` ja `s2` satunnaisesti. Lisäksi määritellään kaksi arvottua kerrointa `k1` ja `k2`, joiden avulla ensimmäisen ja toisen sarakkeen lineaarikombinaatio muodostetaan kolmanneksi sarakkeeksi `s3`. Oletetaan, että järjestelmä arpoo sarakkeeksi `s1` listan `[2,4,5]`, sarakkeeksi `s2` listan `[5,-5,3]`, skalaarin `k1` arvoksi -2 ja skalaarin `k2` arvoksi 2. Tällöin sarakkeen `s3` arvoksi muodostuu `[6,-18,-4]`. Määrittelemämme matriisi `m1` on siis

$$\begin{bmatrix} 2 & 5 & 6 \\ 4 & -5 & -18 \\ 5 & 3 & -4 \end{bmatrix}$$

4.4 STACK ja tietokoneavusteinen arviointi

STACK-projektin tavoitteena on tarjota käytännöllinen ja luotettava arviointijärjestelmä, jossa opiskelija syöttää vastauksen matemaattisena lausekkeena (Sangwin, 2012). Automaattista tarkistamista käytetään tämän vastauksen testaamiseen vertaamalla täyttävätkö kaksi annettua syötettä vaaditut matemaattiset kriteerit. Se siis vertaa opiskelijan vastausta mallivastaukseen. Yleensä verrataan ovatko annettu syöte ja mallivastaus samat, eli ovatko ne algebrallisesti ekvivalentit. Toinen syötteistä on opiskelijan (`SAns`) ja toinen opettajan tai muun tehtävän laatijan muodostama (`TAns`), ja näiden syötteiden erotuksen sievennettyä muotoa

```
simplify(SAns - TAns)
```

verrataan nollaan. Tämä vastauksentarkistus on useimmiten riittävä. Kannattaa huomata, että opettajan vastausta voidaan STACK-järjestelmässä muokata symbolisella ohjelmistolla täyttämään oikean vastauksen kriteerit esimerkiksi satunnaisuutta sisältävissä tehtävätyypeissä. Samoin voimme tarkastella ja muokata opiskelijan vastausta ennen kuin suoritamme arvioinnin.

Lähestulkoon kaikki vastauksen testaamiseen käytettävät funktiot kutsutaan Maximasta. Kaikissa tapauksissa kutsuttavalle funktiolle välitetään seuraavat tiedot; *oppilaan vastaus (Studen-*

tAnswer), opettajan vastaus (TeacherAnswer), sekä valinnainen optio (Opt). Oppilaan ja opettajan vastaukset ovat CAS-muotoisia ilmaisuja vastauksesta. Opettajan vastaus toimii mallivastauksena, johon oppilaan vastausta verrataan. Valinnainen optio voi sisältää jonkin tiedon, jota vastauksen testaamisessa tarvitaan. Tämä voi olla esimerkiksi muuttuja, numeerinen arvo vertailua varten tai merkitsevien numeroiden lukumäärä. (STACK Documentation, Answer tests, 2016)

Vastauksentestausfunktio muodostaa näistä arvoista tuloksen, joka sisältää seuraavat tiedot; *virheet (Errors), tuloksen (Result), palautteen (Feedback) sekä huomautuksen (Note)*. Virhetieto sisältää mahdolliset virheet. Mahdolliset tuloksen arvot ovat **true**, **false**, ja **fail**. Palaute on CASText-muotoinen merkkijono, joka näytetään opiskelijalle. Palaute voidaan muokata sopivaksi riippuen opiskelijan vastauksesta. Huomautusta käytetään raportointiin, eikä se näy opiskelijalle.

```
[Errors, Result, FeedBack, Note] = AnswerTest(StudentAnswer, TeacherAnswer,
    Opt)
```

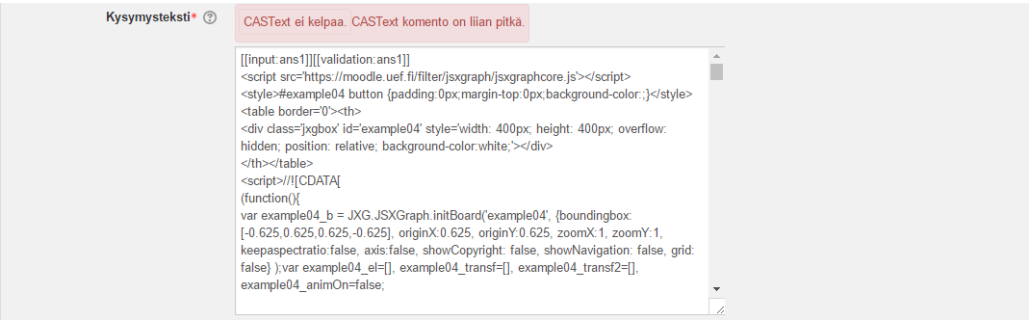
Listaus 9: Vastauksentestaus-funktion syntaksi

STACK-järjestelmän vastauksentestausfunktion muodostaman palautteen lisäksi on myös opettajan usein tarpeen kirjoittaa CAS-muotoista palautetta tehtäviin (Sangwin, 2012). Jotta palaute olisi mahdollisimman oppimista edistävää, on opettajan hallittava hyvin järjestelmän ominaisuudet.

4.5 STACK-järjestelmän rajoitukset

STACK-järjelmä sopii huonosti joillekin kysymystyypeille. Varsinkin todistukset ja kysymykset, joissa halutaan opiskelijoiden kirjoittavan ratkaisu vaihe vaiheelta, ovat haastavia toteuttaa järjestelmällä. Lisäksi muidenkin kysymystyyppien laatimisessa täytyy olla huolellinen, sillä järjestelmä ei vertaa ovatko kaksi satunnaistettua tehtävää vaikeustasoltaan samankaltaisia. Jos tehtävän muuttujia satunnaistetaan, on käyttäjän vastuulla, että kaksi saman tehtävän versiota kysyvät samaa matemaattista kontekstia.

Vastausten syöttäminen on järjestelmässä opiskelijan vastuulla. Vaikka järjestelmä osaakin hyvin huomauttaa virheellisestä (järjestelmälle kelpaamattomasta) ilmaisutavasta vastausta syötettäessä, on lopulta kysymykseen vastaavan tehtävä tarkistaa oma syötteensä. STACK voi vain rajoitetussa määrin korjata käyttäjän syötteen virheitä.



Kuva 12: Liian pitkä CASText-komento

Tutkimuksen aikana havaitsin järjestelmän rakenteessa erään rajoituksen, joka oli kysymystekstikentän syötteen pituuden rajoitus (Kuva 12). Tämä ei luultavasti ole STACK-järjestelmän rajoitus, mutta estää kuitenkin liian pitkän kysymystekstin kirjoittamisen. Toisaalta kysymystekstin täytyy olla hyvin pitkä ennen kuin raja tulee vastaan. Kysymysten laadinnan työvaiheissa tämän ongelman kohtasin pariin otteeseen, ja molemmilla kerroilla kyseessä oli JSXGraph-kuvion pitkästä koodista johtuva tekstin pituus. Tähän ongelmaan on tiettävästi tulossa korjaus syksyllä 2017.

Järjestelmä on pyritty tekemään helppokäyttöiseksi. Opiskelua kuitenkin vaaditaan, että kysymysten laatiminen onnistuu. Osaaminen korostuu siinäkin, että kysymysten toimivuuteen voidaan luottaa. Toimivuus taataan osaamisen lisäksi riittävän kattavalla testauksella, joka sekin on käyttäjän (kysymysten laatijan) vastuulla.

5 STACK-kysymysten konstruoiminen

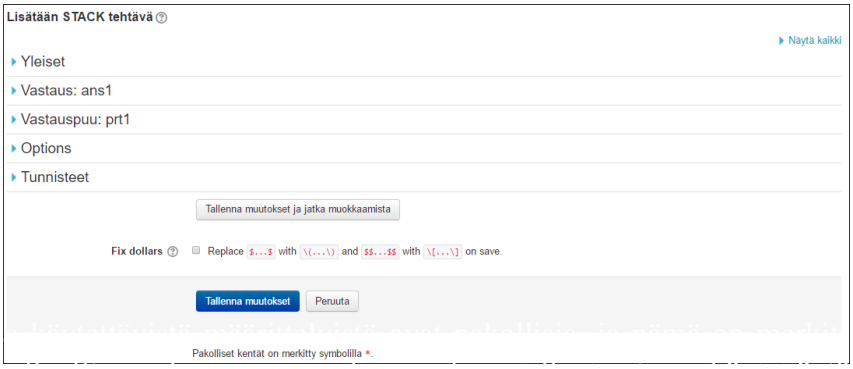
Tässä luvussa käydään läpi STACK-kysymyksen laatimisen vaiheet ja kuvataan asioita, joihin kannattaa kiinnittää erityistä huomiota. Työkokeiluni aikana muodostettiin useita STACK-kysymyssarjoja, joissa kokeilin erilaisin keinoin hyödyntää sekä JavaScript-kieltä että JSXGraph-pohjaisia kuvioita. Tähän olen myös koonnut työni vaiheita STACK-kysymyksiä laatiessani.

STACK-projektin eräs tärkeä tavoite on, että opettajat pystyvät itse laatimaan omia STACK-kysymyksiään (Sangwin, 2015). Luku 5 toimiikin eräänlaisena oppaana STACK-kysymysten ja myös JSXGraph-kuvioita sisältävien STACK-kysymysten formuloimiseen.

5.1 STACK-kysymyksen laatimisen vaiheet

STACK-kysymykset laaditaan Moodle-oppimisympäristössä verkkopohjaisella käyttöliittymällä. Kurssin kysymyspankissa siirrytään kategoriaan, johon kysymys halutaan lisätä ja painetaan Luo uusi kysymys -painiketta. Kysymystyypiksi valitaan STACK. Tehtävänlaadintanäkymään aukeaa seuraavat STACK-kysymyksen rakenteet, joiden avulla kysymys määritellään:

- Yleiset
- Vastaus: ans1
- Vastauspuu: prt1
- Options
- Tunnisteet



Kuva 13: STACK-kysymyksen määrittelysivu

Osa kysymyksen laadinnassa käytettävistä määrittelyistä ovat pakollisia, ja nämä on merkitty kysymyksen muokkaussivulle. Sivun alareunassa on kysymyksen tallentamiseen käytettäviä painikkeita. Lisäksi alareunasta löytyy painike esikatselulle, kunhan kysymykseen on asetettu kaikki pakolliset määritykset. Käydään seuraavaksi läpi kaikki kysymyksen laadinnan vaiheet ja tarkastellaan mitä missäkin kentässä kysymyksen laatijan näkökulmasta tarvitaan tietää.

5.1.1 Yleiset

Yleiset välilehti sisältää kysymyksen niin sanotun näkyvän osan kannalta oleellisimmat rakenteet kuten kysymyksen nimen ja kysymystekstin. Tämä osa kysymyksen laadinnassa sisältää seuraavassa luetellut tehtävän määrittelyt.

Kategoria: Tässä voidaan valita kategoria, johon kysymys sijoitetaan.

Kysymyksen nimi: Kysymykselle kannattaa antaa yksilöllinen nimi. Kannattaa huomata, että vaikka kysymyksen nimi myöhemmin vaihdettaisiin, tämä ei luo uutta kysymystä, vaan vain

sen nimi muuttuu. Kysymyksestä voi erikseen luoda kopion uudelle, tai jopa samalle nimelle. Nimivaihdosten ja kopiointien kanssa pitää olla hyvin huolellinen.

Tehtävän muuttujat: Muuttujat kenttään voidaan asettaa kysymyksessä käytettyjä muuttujien arvoja. Näihin muuttujiin voidaan viitata kysymyksen myöhemmissä vaiheessa, esimerkiksi opiskelijan vastausta tarkistettaessa.

Tehtäväryhmä: Tämän avulla voidaan ryhmitellä kysymyksiä samaa satunnaislukugeneraattorin siementä käyttäviksi, mutta yleensä jätetään tyhjäksi.

Kysymysteksti: Tähän kenttään kirjoitetaan kysymysteksti, joka näkyy opiskelijalle. Kysymystekstikentässä käytetään CASText-muotoilua. Kentässä voidaan käyttää Tehtävän muuttujat -kentässä määriteltyjä kysymyksen muuttujia ja kirjoittaa LaTeX-muotoista tekstiä. Kysymysteksti -kentän tulee sisältää myös vastauskenttäelementti `[[input:ans1]]`. Tämä luo kysymykseen tekstikentän vastausta varten. Tekstikenttä luodaan siihen kohtaan Kysymystekstikenttää, johon asetetaan komento `[[input:ans1]]`, missä `ans1` on vastaustiedon sisältävän muuttujan nimi. Tätä muuttujaa käytetään vastauksen tarkistamiseen. Muuttujalle `ans1` voidaan suorittaa STACK-Maxima -järjestelmän laskutoimituksia tai funktioita ennen tehtävän tarkistamista. Tämän muuttujan nimen voi halutessaan vaihtaa. Myös syötteen esikatseluun tarvittava elementti `[[validation:ans1]]` täytyy syöttää Kysymysteksti-kenttään. Se muodostaa tehtävään esikatseluruudun, joka (halutessa) näyttää opiskelijalle kuinka hänen vastauksensa tulkitaan järjestelmässä. `input` ja `validation` ovat tehtävän interaktiivisia elementtejä, joiden avulla opiskelija voi vastata kysymykseen ja tarkastella oman syötteensä kelvollisuutta (Sangwin, 2012). Kuvassa 16 näkyy ylempänä kysymykseen laadittu vastauskenttäelementti, ja alempana syötteen esikatselualue. Kysymykseen voidaan luoda useampi kuin yksi vastauskenttä, jolloin ne täytyy nimetä yksilöllisesti, ja jokaiselle vastauskentälle on luotava myös sitä vastaava esikatselualue vaikkei esikatselua näytettäisikään (Listaus 10). Vastauksen syötekentän *tyyppi* voi olla muukin kuin tekstisyöte. Tähän palataan Vastaus:ans1 -välilehden asetuksissa.

```
Anna lukuvalintojasi vastaava matriisi \(A\).
\(A = \) [[input:ans1]] [[validation:ans1]]
Silloin \(det A = \) [[input:ans2]] [[validation:ans2]]
```

Listaus 10: STACK-kysymystekstiin määritelty kaksi vastauskenttää

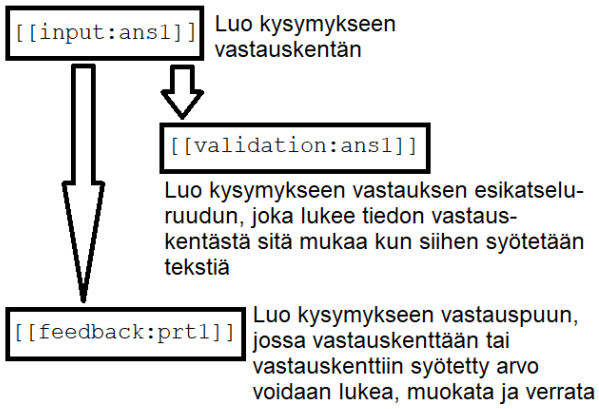
Niin Kysymysteksti- kuin muitakin CASText-kenttiä muokatessa on kentän muotoiluksi vaihdettava HTML-muoto, jotta STACK- ja JSXGraph-syntaksi tulkitaan Moodle-ympäristössä oikein. Editorina on käytettävä Pelkkä teksti -editoria (Plain text), jonka voi käydä muuttamassa Moodlen omista asetuksista. Kuvassa 14 HTML-muoto on vaihdettu, ja tekstieditorina on Pelkkä teksti.

Oletuspisteet: Tähän syötetään kysymyksen antamat oletuspisteet. Oletusarvo on 1, jota useimmiten ei tarvitse muuttaa; kysymysten painoarvoa voi halutessaan säätää Moodlen tenttityökalussa.

Erityinen palaute: Tämä osio sisältää oletusarvoisesti elementin `[[feedback:prt1]]`, joka luo kysymykseen vastauspuun nimeltä `prt1`. Vastauspuussa suoritetaan vastauksen vertaaminen oikeaan vastaukseen ja näytetään palaute opiskelijalle. Kyseinen `[[feedback:prt1]]`-elementti voidaan siirtää niin halutaessa myös kysymystekstiosioon, jolloin vastauspuun palaute näytetään siinä paikassa kysymystekstiä, johon tämä elementti on asetettu. Palaute-elementtejä ei tarvitse määritellä yhtä monta kuin vastauskenttiä on määritelty, vaan yhdessä palaute-elementissä voidaan käsitellä useamman vastauskentän syötteitä. Usein on kuitenkin selkeämpää

Kuva 14: HTML-muotoilu CASText-kentissä

asettaa jokaiselle vastauskentälle `[[input]]` oma palaute-elementtinsä, jossa kyseiseen vastauskenttään syötetty vastaus tarkistetaan. Jokaisessa vastauspuussa voidaan käsitellä kaikkien vastauskenttien syötteitä.



Kuva 15: Syötekentän tieto luetaan esikatseluruutuun ja tehtävän vastauspuuhun

Rangaistus: Pistemäärä, joka vähennetään jokaisen väärän yrityskerran jälkeen.

Yleinen palaute: Tähän voidaan syöttää kysymyksen mallivastaus CASText-muodossa. Palaute näytetään opiskelijalle sitten, kun hän on vastannut kysymykseen. Tässä palautteessa voidaan käyttää Tehtävän muuttujat -kentässä määriteltyjä muuttujia. Yleinen palaute voi sisältää esimerkiksi mallin kysymyksen ratkaisun vaiheista.

Tehtävän erotteluteksti: Tällä kentällä erotellaan saman kysymyksen eri versiot toisistaan, kun kysymys sisältää satunnaisia muuttujia. Koska satunnaisuutta sisältävät kysymykset ovat muuttujien arvoiltaan erilaisia, tämän kentän avulla satunnaistetut kysymykset erotellaan. Tähän kenttään täytyy siis laittaa ainakin yksi satunnainen muuttujan arvo, mutta siihen kannattaa laittaa jokainen satunnaisuutta sisältävä muuttuja.

5.1.2 Vastauskentän määrittely (Vastaus:ans1)

Seuraava välilehti on Vastaus:ans1, joka nimensä mukaisesti sisältää kysymyksen vastauskenttään liittyviä määrittelyjä. Nimi *ans1* riippuu kysymyksen vastauksena käytettävän muuttujan tai muuttujien nimestä, ja vastauksen tyyppi voidaan vaihtaa tässä. Oletusarvoisesti vastaustyyppi on tekstikenttä, johon opiskelija kirjoittaa vastauksensa matemaattisena ilmauksena, mutta tässä osiossa myös vastaustyyppi on mahdollista vaihtaa esimerkiksi matriisiksi. Myös syötteen tarkastamiseen liittyviä optioita voi määritellä tällä välilehdellä. Kysymys voi sisältää yhden sijasta useita vastauskenttiä, jolloin jokaiselle vastauskentälle luodaan omat määrittelyt.

Vastauksen tyyppi on oletusarvoisesti algebrallinen lauseke. Se muodostaa vastauskentäksi tekstikentän, johon vastaus (syöte) voidaan kirjoittaa Maxima-syntaksilla. Vastauksen tyyppiä voidaan myös muuttaa. Seuraavassa on listattu mahdolliset vastaustyypit STACK-järjestelmässä (STACK Documentation, 2016):

- algebrallinen lauseke (algebraic input),
- valintaruutu (checkbox),
- matriisi (matrix),
- oikein/väärin (true/false),
- pudotusvalikko (drop down list),
- valintanappi (radio button),
- tekstialue (text area),
- yksikkö (units)
- yksittäinen merkki (single character).

Mallivastaus on opettajan määrittelemä oikea ratkaisu kysymykseen. Tämä kenttä voi sisältää kysymyksessä määriteltyjä muuttujia. Tähän asetettua syötettä ei kuitenkaan käytetä vastauksen oikeellisuuden tarkistamiseen, vaan se tapahtuu myöhemmin kohdassa Vastauspuut. Tämä on kuitenkin pakollinen kenttä, koska joissakin tapauksissa järjestelmä tunnistaa tähän syötetystä arvosta, millainen kyseisen vastauskentän tulee olla rakenteeltaan. Esimerkiksi, jos vastauskentäksi halutaan matriisi, on tähän syötettävä mallivastauksena dimensioltaan samankokoinen matriisi kuin vastausmatriisiksi halutaan.

Vastauskentän pituus määrittelee tekstikentän pituuden kysymyssivulla. Oletusarvoisesti pituus on 15 merkkiä.

Maxima-syntaksi määrittelee sen, vaaditaanko vastauksen olevan Maxima-syntaksin mukaista.

Lisätäänkö tähdet -määrittelyllä voidaan asettaa STACK-järjestelmä lisäämään kertomerkki (*) opiskelijan syöttämään lausekkeeseen. Oletusarvoisesti tämä määrittely on pois päältä, ja opiskelijoiden onkin hyvä oppia kirjoittamaan vaaditun syntaksin mukaisesti. Mikäli tätä optiota kuitenkin käyttää, on valittavana kaksi vaihtoehtoista tapaa kertomerkkien lisäämiselle (STACK Documentation, 2016):

- lisää tähdet epäsuoralle kertolaskulle (*insert stars for implied multiplication*): lisää kertomerkit lausekkeeseen, mikäli JavaScriptin Strict[22]-syntaksi tulkitsee kertomerkkien puuttumisen.

[22] `https://www.w3schools.com/js/js_strict.asp`

- lisää tähdet oletuksella, että käytetään yhden merkin muuttujia (*insert stars assuming single-character variable names*): lisää kertomerkit lausekkeeseen sillä oletuksella, että opiskelija ilmaisee muuttujat yhdellä merkillä. STACK tunnistaa ja erottelee syötteestä muuttujat ja funktiot. Esimerkiksi syöte `sin(ax)` tulkitaan muodossa `sin(a*x)`, ei muodossa `s*i*n*(a*x)`.

Syntaksivihje asettaa kysymyksen vastauskenttään vastausmallin mikäli niin halutaan. Syntaksivihje toimii eräänlaisena mallina siitä, missä muodossa vastaus täytyy syöttää. Esimerkiksi jos vastauksena on syötettävä lista, voidaan tähän asettaa malliksi vaikkapa `[?,?]`

Kielletyt merkkijonot kentässä voidaan kieltää käyttämästä joitakin sanoja vastauskentässä. Tämä voi olla viisasta varsinkin, jos opiskelijoissa on vahvoja Maxima-syntaksin osaajia, jolloin opiskelija voisi esimerkiksi ratkaista yhtälön `solve`-komennon avulla ratkaisematta yhtälöä itse. Voisimme siis tässä tapauksessa kieltää sanan 'solve' käytön.

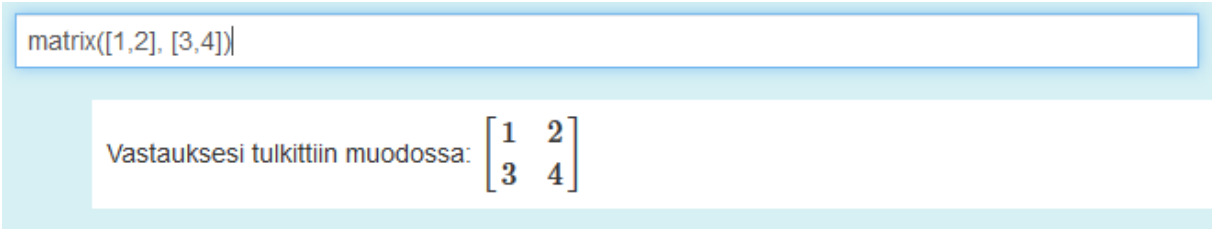
Sallitut sanat: Tässä voidaan sallia mielivaltaisten funktioiden tai yhtä merkkiä pidempien muuttujien käyttö lisäämällä poikkeukset tähän kenttään.

Liukuluvut kielletään -valinta on oletusarvoisesti 'Kyllä'. Mikäli liukuluvut on kielletty, eli vaaditaan tarkka arvo, ja opiskelija vastaa liukulukusyötteellä, vastaus hylätään.

Vaaditaanko supistettu muoto asettaa vaatimuksen, että vastaukseksi syötettävän murtoluvuvun on oltava supistetussa muodossa. Oletusarvoisesti tämä valinta on 'Ei'.

Tarkista vastauksen tyyppi -valinnalla asetetaan järjestelmä tarkistamaan, että opiskelijan vastaus on samaa muotoa kuin opettajan vastaus. Tämä on hyödyllinen esimerkiksi siinä tapauksessa, että opiskelija saattaa syöttää yhtälön muodossa $y = mx + c$ eikä ilmaisuna $mx + c$, koska niitä ei voida tarkistaessa suoraan verrata toisiinsa. Toinen keino varmistaa opiskelijan syötteen tyyppi tämän kaltaisessa tehtävässä on kirjoittaa tehtävänantoon $y =$ ennen vastauskenttää, eli `\( y = \) [[input:ans1]]`. (STACK Documentation, 2016).

Esikatselu -valinnalla voidaan määrätä, näytetäänkö opiskelijan vastaus tämän vastauskentän kohdalla esikatselussa. Tämä on useimmiten hyvä olla valittuna, jotta opiskelija voi varmistua missä muodossa järjestelmä hänen vastauksensa tulkitsee. Kuvassa 16 näkyy kuinka tämä käytännössä toimii.



Kuva 16: Vastauksen esikatselu

Näytä validointi optiolla voidaan asettaa opiskelijoille näkyvä palaute kysymyksessä. Myös kysymyksen virheilmoitukset ovat osa kysymyksestä annettavaa palautetta. Näytä validointi voi olla myös 'Ei', jolloin palautetta ei näytetä, mutta muussa tapauksessa tämän määrittelyn avulla voidaan näyttää palautetta opiskelijan syötteestä, muuttujista ja yhdistää mahdolliset vastauspuiden antamat palautteet.

Extra options kenttään voidaan syöttää muunlaisia vastaustyyppejä kuin mitä STACK tarjoaa, mutta niiden tulee olla CAS-syntaksin mukaisia (STACK Documentation, 2016). Tämä on lähinnä järjestelmäkehittäjille tarkoitettu kenttä.

5.1.3 Vastauksen tarkistaminen ja vastauspuut (Vastauspuu:prt1)

Tässä kysymyksen laatimisen osiossa muotoillaan vastauspuu, joka sisältää kysymyksen tarkastamiseen liittyvät elementit. Myös kysymyksen antama palaute muotoillaan tässä osiossa ja rakennetaan vastauspuu kysymyksen antamasta palautteesta. Hyvin laaditun vastauspuun avulla järjestelmällä on mahdollista antaa kattavaa formatiivista palautetta opiskelijan suoriutumisesta (Sangwin, 2012). Mikäli kysymyksessä on useampi vastauskenttä, voidaan jokaisen vastauksen arviointi suorittaa yhdessä vastauspuussa. Suositeltavaa kuitenkin on määritellä jokaiselle vastakselle oma vastauspuunsa. Tämä siksi, että mikäli johonkin vastauskenttään, jota vastauspuussa käsitellään on jätetty vastaamatta, ei järjestelmä suorita vastauksen arviointia oikein. Lisäksi on helpompaa luoda jokaiselle vastauskentälle oma vastauspuunsa, jossa vastausta arvioidaan.

Kysymyksen arvolla voidaan säätää vastauksen painoarvoa esimerkiksi tilanteessa, jossa kysymys sisältää useamman vastauskentän. Jos kysymyksessä on kaksi vastauskenttää ja molempien vastausten arvoksi on asetettu 1, kohdellaan niitä samanarvoisina ja kummastakin saa 50% tehtävän pisteistä. Huomioitavaa on, että mikäli vastauspuu sisältää enemmän kuin yhden solmun, on solmuista saatavien pisteiden summan syytä olla sama kuin tämä kysymyksen arvo.

Automaattinen sievennys käskee Maximaa sieventämään ilmaisut, joiden avulla vastaus tarkistetaan. Opiskelijan vastauksen ei siis tarvitse olla sievennetyssä muodossa, jos tässä on valittuna ‘Kyllä’.

Palautteen muuttujat -kenttä on CASText-muotoinen ja siinä voidaan käyttää kysymyksen muuttujia tai/ja opiskelijan syötettä laskutoimituksiin. Opiskelijan syötettä voidaan käyttää sillä muuttujannimellä, joka määriteltiin kysymystekstin `input`-kentässä. Esimerkiksi määrittelyllä `[[input:ans1]]`, joka on oletusarvoisesti määritelty uudessa STACK-kysymyksessä, muuttujan `ans1` arvo(j)a voidaan käyttää tässä kentässä. Voimme esimerkiksi syöttää palautteen muuttujaksi boolean-tyyppisen muuttujan `on_oikein`

```
on_oikein: if abs(ans1 - TAns1) < 0.1 then true else false;
```

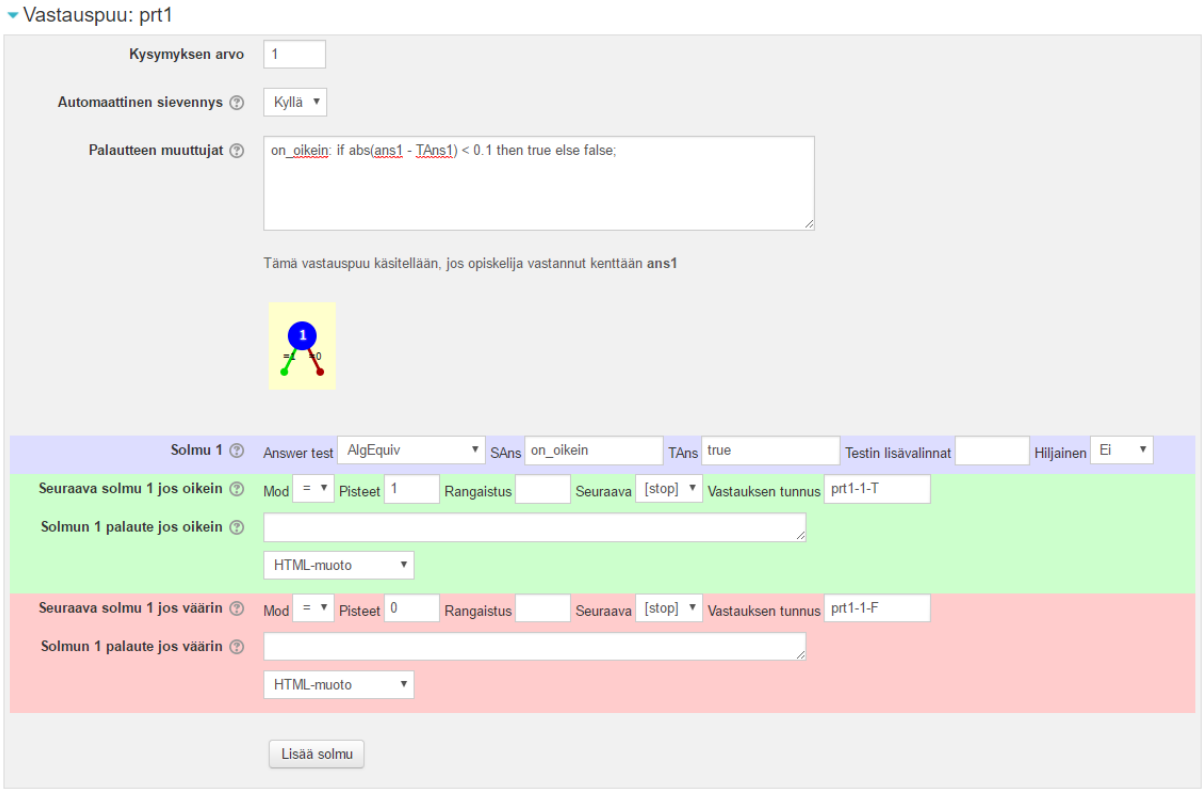
joka saa arvon tosi mikäli arvo `ans1` on riittävän lähellä toivottua arvoa `TAns1`. Tässä tapauksessa eron täytyy olla pienempi kuin 0,1 yksikköä.

Luvussa 5.1.1 mainittiin, että yhden vastauspuun avulla voimme arvioida usean syötteen tuloksia. Vastauskenttien arvot ovat siis käytössä vastauspuiden muuttujakentässä yleisesti. Esimerkiksi määrittely

```
final_ans: if diff(ans1,x) = ans2 then true else false;
```

muodostaa muuttujalle `final_ans` arvon sen mukaan, mitä laskennan tulos muuttujien `ans1` ja `ans2` suhteen tuottaa. Laskennan jälkeen muuttujaa `final_ans` voidaan käyttää tarkastusfunktion syötteenä.

Solmu 1 on vastauksen tarkistuksen osalta se vaihe, jossa opiskelijan vastausta verrataan oikeaan vastaukseen. Tästä voidaan edetä seuraavaan solmuun riippuen siitä mitä vastaukselta vaaditaan. Solmuja on oletusarvoisesti vain yksi, jolloin tehtävä antaa palautetta oikeasta tai väärästä vastauksesta vain tämän solmun määrittelyjen mukaisesti. *Answer test* -valinnalla voidaan



Kuva 17: STACK-kysymyksen Vastauspuu

vaikuttaa siihen, kuinka vastauksia verrataan keskenään. STACK-järjestelmän tavat vertailla vastauksia käsiteltiin luvussa 4.4, ja usein valinta 'AlgEquiv' on hyvä tapa verrata vastauksia keskenään. Työkokeiluni aikana tekemissäni tehtävissä jokaisessa on käytössä tämä tarkastusmetodi.

Kenttään *SAns* syötetään tarkistusfunktion ensimmäinen alkio, joka on opiskelijan vastaus (`ans1`) tai siitä laskemalla tai Maxima-funktiolla muodostettu arvo. Kuvassa 17 arvoksi on asetettu Palautteen muuttujat -kentässä määritelty arvo. Kenttään *TAns* syötetään vastaavasti toinen tarkastusfunktion alkio, johon opiskelijan vastausta verrataan. Kentän TAns arvoksi voi asettaa aiemmin Tehtävän muuttujat -kentässä määrittelemänsä arvon tai jonkin muun CAS-ilmaisun. Valinnalla *Hiljainen* voidaan asettaa näkyykö solmun tuottama palaute opiskelijalle vai ei.

Seuraava solmu 1 jos oikein: Mikäli opiskelijan vastaus oli oikein, suorittaa järjestelmä toiminnot ja seuraavaan solmuun siirtymisen siten, kuin ne on tässä määritelty. Tässä voidaan myös määritellä vastauksesta saatavien pisteiden määrää. *Seuraava*-valikosta voidaan valita mihin solmuun arvioinnissa seuraavaksi siirrytään, mikäli solmuja on määritelty useampi kuin yksi, ja mikäli solmusta halutaan siirtyä toiseen solmuun.

Solmun 1 palaute jos oikein: Tähän voidaan syöttää CASText-kirjoitusta, jossa opiskelijalle annetaan palautetta tässä solmussa arvioidusta oikeasta vastauksesta.

Seuraava solmu 1 jos väärin: Tämä on vastaava kuin Seuraava solmu 1 jos oikein -kohta, mutta suoritetaan kun opiskelijan vastaus arvioidaan tässä solmussa vääräksi.

Solmun 1 palaute jos väärin: CASText-muotoinen tekstikenttä, johon voi kirjoittaa palautteen väärästä vastauksesta tämän solmun arvioinnissa.

Solmuja voi listätä panikkeella ’Lisää solmu’. Kuvassa 18 on kuvattuna solmurakenteen toiminta kysymyksessä, jossa on kaksi solmua.

Kuva 18: Vastauspuu, jossa on kaksi vastausta arvioivaa solmua

5.1.4 Tehtävän muut valinnat (Options)

Tässä valikossa on yleisiä STACK-kysymystä koskevia asetuksia. Useimmiten nämä asetukset voi jättää muuttamatta. *Tehtäväkohtainen sievennys* -valinnalla voidaan muuttaa STACK-järjestelmän tapaa suorittaa vastauksen vertaaminen vastauspuussa. Mikäli tämä optio muutetaan arvoksi 'Ei', järjestelmä ei sievennä vastauksia vastauspuissa. *Oletetaan positiiviseksi* -valinnalla voidaan järjestelmä asettaa olettamaan että vastaus on positiivinen.

Yleinen palaute oikean vastauksen jälkeen, *Yleinen palaute osittain oikean vastauksen jälkeen* ja *Yleinen palaute väärän vastauksen jälkeen* -kentissä kysymykseen voidaan luoda yleinen palaute, joka opiskelijalle näytetään riippuen hänen vastauksensa oikeellisuudesta. Näihin kenttiin voidaan syöttää CASText-muotoista palautetta. Näiden palautekenttien teksti näytetään opiskelijalle aina, vaikka vastauspuussakin olisi määritelty palautetekstit. Mikäli kysymykseen on määritelty sekä yleinen palaute että vastauspuun palaute, näytetään opiskelijalle molemmat.

Kertomerkki -valinnalla voidaan asettaa CASText-muotoilussa näytettävän kertomerkin esitystapa. Oletuksena järjestelmä asettaa kertomerkin esitystavaksi 'Ei yhtään', jolloin kertomerkkiä ei näytetä ilmaisuissa, ja esimerkiksi CASText `\(3*x\)` tulostuu tekstiin muodossa $3x$. Kertomerkin esitystavaksi voidaan valita 'Risti', jolloin ilmaisu tulostuu muodossa $3 \times x$ tai 'Piste', jolloin ilmaisu tulostuu muodossa $3 \cdot x$.

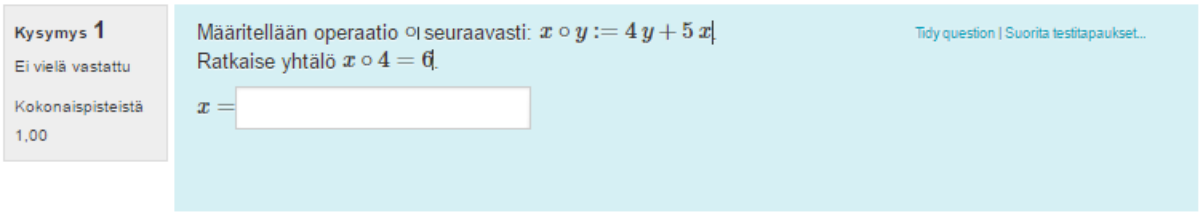
Juurien esitystapa -valinnalla voidaan muuttaa määritystä, kuinka juurilausekkeet ilmaistaan CASText-muotoilussa. Mikäli asetus on 'Kyllä' pyritään näyttämään juurilausekkeina kysymyksessä määritellyt muuttujat tai järjestelmän laskemat arvot, jotka ovat murtopotensseja. Valinnalla *Ilmaisun sqrt(-1) tulkinta ja esitys* voidaan asettaa symbolien `i` ja `sqrt(-1)` tulkinta- ja esitystapa. *Inverse trigonometric functions* -valinnalla voidaan asettaa arcus-funktioiden esitystapa CASText-kentissä. *Default shape of matrix parentheses* -valikosta voidaan valita millainen sulkumerkintä matriiseille tulostetaan CASText-muotoilussa. Valittavana on myös sulkujen poisto.

Vihje -kenttiin voidaan kirjoittaa vihje, joka näytetään opiskelijalle väärän vastauksen jälkeen mikäli kysymystä käytetään tentissä, jota voi yrittää tehdä useammin kuin yhdesti. Vihje 1 -kenttään kirjoitetaan palaute, joka näytetään ensimmäisen suorituskerran jälkeen, Vihje 2 -kenttään toisen suorituskerran vihje jne. Vihje-kenttiä voi lisätä painikkeesta 'Lisää vihje'. Vihje-kenttiin voidaan kirjoittaa CASText-muotoista tekstiä.

5.1.5 Esimerkkikysymys

Tässä kuvailen erään syksyllä 2016 kurssille Matematiikan johdantokurssi tekemäni STACK-kysymyksen. Kuvassa 19 on esikatselukuva valmiista kysymyksestä.

Kysymyksessä on määritelty satunnaisia muuttujia, jotka ovat laskutoimituksessa $x \circ y := 4y + 5x$ esiintyvät arvot 4 ja 5, sekä varsinaisen kysymyksen 'Ratkaise yhtälö $x \circ 4 = 6$' arvot 4 ja 6. Nämä satunnaiset arvot muodostetaan Tehtävän muuttujat -kentässä Listauksen 11 mukaisesti.



Kuva 19: Laskutoimitus-STACK -kysymyksen esikatselu

```
b: rand([-4,-3,3,4]);
c: rand([-6,-4,-2,2,4,6]);
d: rand([-4,-2,2,4]);
a: if mod(d-b*c,3)=0 then (d-b*c)/rand([-3,3]) else if mod(d-b*c,4)=0 then
    (d-b*c)/rand([-4,4]) else (d-b*c)/rand([-2,2]);
xoy: a*x+b*y;
TAns1: rhs(solve([a*x + b*c - d = 0],[x])[1]);
```

Listaus 11: Laskutoimitus-kysymyksen Tehtävän muuttujat -kenttä

Tehtävän muuttujat -kentässä määritellään ensin kolme muuttujaa `b`, `c` ja `d` tietyin rajoituksin satunnaisiksi. Kysymyksestä haluttiin rakentaa sellainen, että sillä on aina kokonaislukuratkaisu, jota ratkaistaessa laskettavat lukuarvot eivät ole liian suuria päässälaskettavaksi. Tässä voimme varmistua siitä rajoittamalla muuttujien valintaa. Tahdomme nyt, että muuttujan `a` arvoksi ei tule 0 millään muilla muuttujien arpomisilla. Siinä tapauksessa kysymyksessä esitetty laskutoimitus olisi x:stä riippumaton, ja kysymys muuttuisi triviaaliksi. Varmistetaan siis ensin, ettei laskutoimituksen `d-b*c` tulos ole 0 valitsemalla muuttujat sopivasti.

Nyt voisimme muodostaa muuttujalle `a` arvon esimerkiksi asettamalla

```
a: (d-b*c) / 2.
```

Tällöin tehtävänanto on kyllä aina satunnaistettu, mutta tehtävän ratkaisuksi muodostuisi aina $x = 2$. Määrittelemällä `a` ehtolauseita käyttäen kuten tässä, saamme kysymyksen ratkaisuksi kuusi eri vaihtoehtoa $x = -4$, $x = -3$, $x = -2$, $x = 2$, $x = 3$ ja $x = 4$.

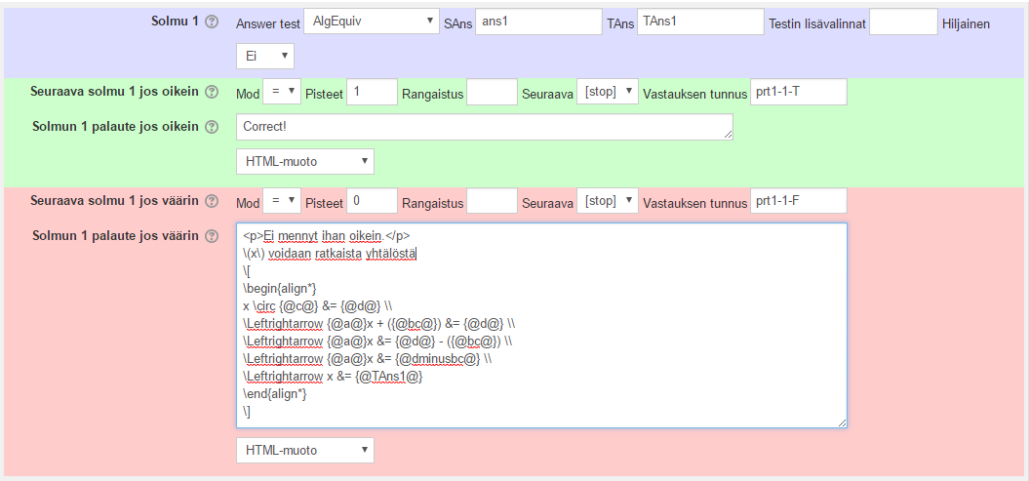
Kysymysteksti-kenttään kirjoitetaan Listauksen 12 mukainen CASText. Siinä näytetään Tehtävän muuttujat -kentässä määriteltyjä muuttujia `xoy`, `c` ja `d` osana tehtävänantoa. Kysymykseen luodaan vastauskenttä `[[input:ans1]]` ja vastauksen muodon esikatseluruutu `validation`. Vastauskentän eteen kirjoitetaan selvennykseksi $x =$, jotta voidaan olla varmempia siitä, ettei opiskelija syötä itse vastausta muodossa `x= vastaus`. Erityinen palaute -kenttää ei muuteta, vaan siinä on oletusarvoisesti `[[feedback:prt1]]`.

```
<p>
Määritellään operaatio \( \circ \) seuraavasti: \( x \circ y := {@xoy@} \).
<br>
Ratkaise yhtälö \( x \circ {@c@} = {@d@} \).
<p/>
<p>\( x= \)[[input:ans1]] [[validation:ans1]]</p>
```

Listaus 12: Laskutoimitus-kysymyksen Kysymysteksti-kenttä

Vastauskentän `ans1` asetukset on määritelty kokonaan järjestelmän oletusarvojen mukaisesti. Mallivastaukseksi on asetettu Tehtävän muuttujat -kentässä määritelty muuttuja `TAns1`. Vastauspuu:prt1-osiossa tarkistetaan opiskelijan vastaus ja rakennetaan palautetta opiskelijalle väärästä vastauksesta. Palautteen muuttujat -kenttään on muodostettu kaksi muuttujaa, joita käytetään palautteen antamiseen:

```
bc: b*c;
dminusbc: d-bc;
```



Kuva 20: Laskutoimitus-STACK-kysymyksen Solmu 1 -määrittelyt

Vastauspuun Solmu 1 on määritelty Kuvan 20 mukaisesti. Vastaus testataan 'AlgEquiv'-metodilla, ja verrattavat ilmaisut ovat opiskelijan syöte `ans1` ja määrittelemämme `TAns1`. Väärästä vastauksesta muodostetaan palaute, jossa ratkaisu esitetään vaihe vaiheelta. Muilta osin tehtävän määrittelyt ovat oletusarvoiset.

5.2 JSXGraph kuvion liittäminen kysymystekstiin

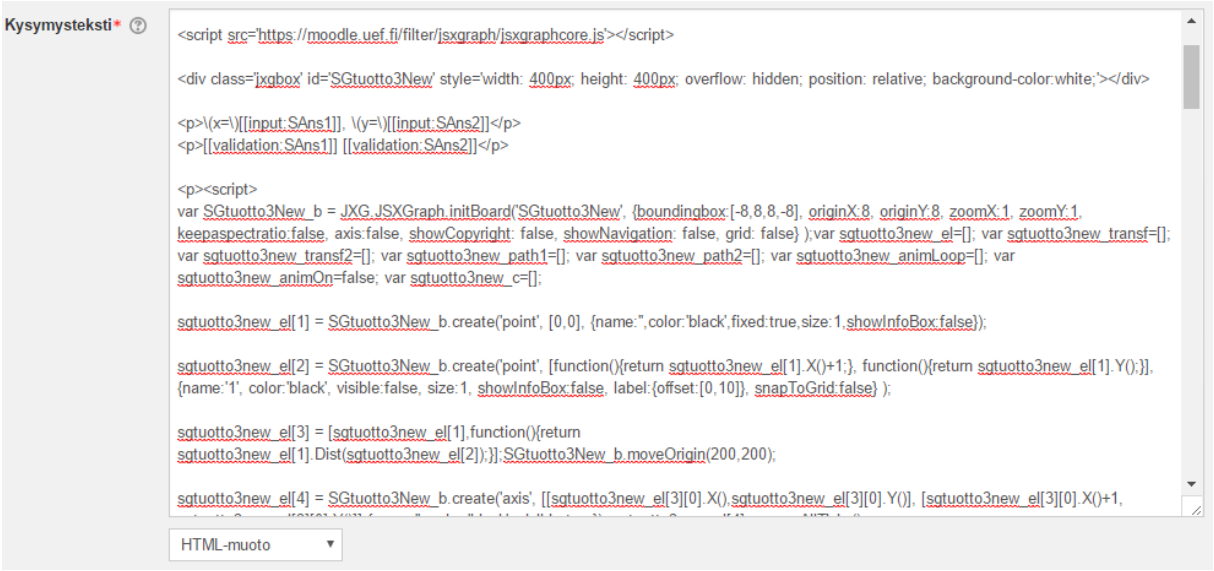
Ennen kuin kuvio voidaan liittää kysymykseen, on kysymystekstiin linkitettävä JSXGraph-ohjelmakirjasto (*Framework*). Itä-Suomen yliopistolla JSXGraph löytyy osoitteesta `https://moodle.uef.fi/filter/jsxgraph/jsxgraphcore.js`. Ohjelmakirjaston voi ladata lisäämällä kysymystekstiin rivin

```
<script src='https://moodle.uef.fi/filter/jsxgraph/jsxgraphcore.js'>
                                    </script>
```

joka linkittää ohjelmakirjaston kysymyksen www-sivuun.

JSXGraph-kuvion liittäminen kysymykseen onnistuu lisäämällä kysymystekstiin `<div>`-tagin, jonka tunniste on kuvion nimi. `<div>`-tagi muodostaa kysymyksen HTML-dokumenttiin kuva-alueen JSXGraph-kuviota varten. Tehtävään voidaan myös ladata JSXGraph-muotoiluasetukset

(kts. Luku 3.3), mutta usein on riittävää suorittaa muotoilu itse. Yleensä riittää määritellä muotoiluasetukset `overflow` ja `position` kuva-alueen määrittelyn yhteydessä (Listaus 13, rivi 2), mutta aina nämäkään eivät ole välttämättömiä. `<div>`-tagin sijainnilla kysymystekstissä voimme vaikuttaa kuvion sijaintiin lopullisessa kysymyksessä. Tämän lisäksi Kysymystekstikenttään lisätään kuvion muodostava JavaScript-koodi. Kuvassa 21 näkyy kuinka `<div>`-tagi ja JavaScript-koodi on sijoitettu.



Kuva 21: JSXGraph-kuvion liittäminen kysymykseen

Kuvioon voidaan halutessa viedä Tehtävän muuttujat -kentässä määriteltyjä muuttujia. Tämä voi olla mielekästä esimerkiksi sellaisissa tilanteissa, joissa kuvion lähtötilannetta halutaan satunnaistaa. Tehtävän muuttujat -kentässä määriteltyjen muuttujien arvot viedään Kysymystekstikenttään kirjoittamalla ne muodossa `{#muuttujan_nimi#}`, jotta JavaScript tulkitsee ne oikein. Olkoon muuttuja `P` määritelty Tehtävän muuttujat -kentässä seuraavasti:

```
P: rand( [ [3,3], [2,4], [-2,4], [-3,3] ] );
```

Listauksessa 13 on esitetty, kuinka satunnaistetun muuttujan `P` arvo viedään kuvioon muuttujan `piste` lähtöpisteeksi.

```
1 <script src='https://moodle.uef.fi/filter/jsxgraph/jsxgraphcore.js'>
    </script>
2 <div class="jxgbox" id="Taso" style="width: 400px; height: 400px;
    overflow: hidden; position: relative;"></div>
3 <script>
4 var taulu = JXG.JSXGraph.initBoard('Taso', {boundingbox: [-5,5,5,-5],
    keepaspectratio: false, axis: true, showCopyright: true,
    showNavigation: false, grid: true} );
5 var piste = taulu.create('point', {#P#}, {name:'P'});
6 </script>
```

Listaus 13: STACK-muuttujan käyttäminen JSXGraph-kuviossa

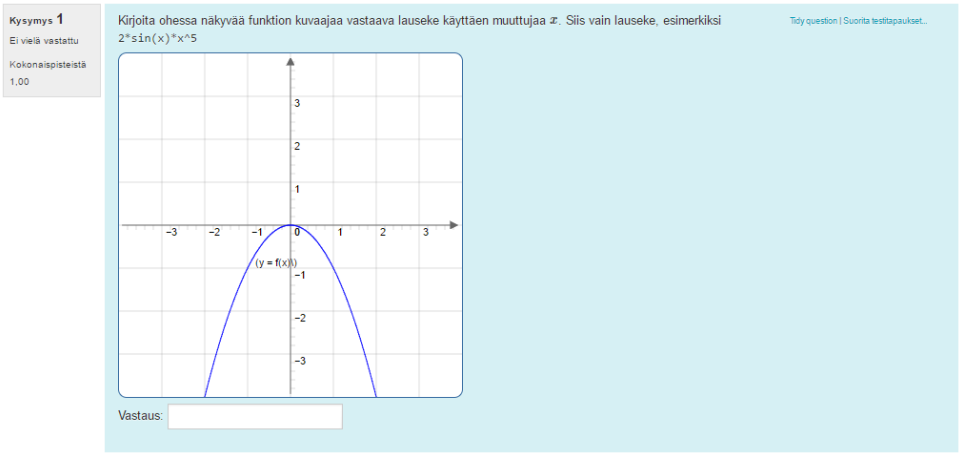
Tämä ominaisuus on hyödyllinen myös silloin, kun opiskelija on vastannut tehtävään ja haluamme näyttää kuvion avulla ominaisuuksia, joita hänen vastauksensa tuottaa. Voimme poimia vastauskentästä opiskelijan syöttämän vastauksen arvon ja käyttää sitä sellaisenaan tai muokattuna kuvion rakentamiseen. Tästä myöhemmin Luvussa 5.3.

5.2.1 Esimerkkikysymys: JSXGraph-kuvion liittäminen STACK-kysymykseen

Tässä kysymyksessä opiskelijalle näytetään funktion kuvaaja, jota vastaavaa lauseketta häneltä kysytään. Kuvio 22 kysymyksen esikatseluikkunasta havainnollistaa kysymyksen asettelua ja kuviota. Tehtävässä haluttiin luoda satunnainen funktio ja sen kuvaaja JSXGraph-kuvioon. Funktion kuvaaja voidaan piirtää JSXGraph-komennolla `functiongraph`, esimerkiksi

```
board.create('functiongraph', [function(x)return 2*x]);.
```

JavaScript- ja Maxima-ilmaisut poikkeavat kuitenkin jossain määrin toisistaan. STACK-järjestelmässä määritelty lauseke `x^2` ei toimi sellaisenaan JSXGraph-komennossa käytettynä. JSXGraph-funktiolle ilmaisu täytyy kirjoittaa muodossa `Math.pow(x,2)`. Piirrettävän kuvaajan määritteleminen satunnaisesti on siksi hieman mutkikkaampaa. Listauksessa 14 näkyvät kysymyksen Tehtävän muuttujat -kentän määrittelyt. Riveillä 1 ja 2 on määritelty JSXGraph-kuvion koordinaatiston koko ja kuvion tauluelementille yksilöllinen nimi. Satunnaisen lausek-



Kuva 22: STACK-kysymys, jossa on käytetty JSXGraph-kuviota

keen määritteleminen on ratkaistu seuraavasti:

- Rivillä 5 arvotaan luku väliltä 1-11.
- Rivillä 6 määritellään lista merkkijonoina asetettuja lausekkeita, jotka ovat JavaScript-syntaksille kelvollisia.
- Rivillä 7 määritellään toinen lista lausekkeita, joiden järjestys ja matemaattinen merkitys vastaa rivin 6 listan lausekkeita, ja jotka ovat Maxima-syntaksille kelvollisia.
- Poimitaan molemmista listoista lauseke indeksistä, joka arvottiin rivillä 5.

```
1  boardsize: [-4, 4, 4, -4];
2  boardname: "F1PGS01_Stack";
3
4  /*arvotaan satunnainen luku (indeksi)*/
5  randf: rand([1,2,3,4,5,6,7,8,9,10,11])
6  /*luodaan kaksi listaa, jossa saman merkityksen omaavat termit sijaitsevat
       samassa listan indeksissä*/
7  fx_js_array:["2*x", "-x", "-2*x", "3*x", "-3*x", "1-x", "x-1", "-1*Math.pow
       (x,2)", "Math.pow(x,3)", "Math.pow(x,2)-2", "-1*Math.pow(x,2)+1"];
8  fx_ans_array: [2*x, -x, -2*x, 3*x, -3*x, 1-x, x-1, -x^2, x^3, x^2 - 2, -x^2
       + 1];
9
10 /*poimitaan listoista samaa tarkoittavat termit, toinen javascriptille ja
       toinen stackille */
11 fx_js:fx_js_array[randf];
12 fx_ans:fx_ans_array[randf];
13 TAns1 : fx_ans;
```

Listaus 14: Tehtävän muuttujat -kenttä esimerkissä JSXGraph-kuvion liittäminen kysymykseen

Näin meillä on kaksi ilmaisua samasta lausekkeesta, joista toinen on kelvollinen Maxima-ilmaisu ja toinen JavaScript-ilmaisu. Kysymysteksti-kenttä on esitetty Listauksessa 15. JSXGraph-taulun nimi määritellään riveillä 7 ja 12 STACK-muuttujan `boardname` mukaan. Rivillä 12 määritellään myös taulun koko. Rivillä 13 luodaan kuvioon funktion kuvaaja satunnaistamallamme lausekkeella. Koska lauseke on tyyppiä merkkijono, täytyy se JavaScript-koodissa suorittaa `eval()`-funktiolla, joka laskee tai suorittaa merkkijonon mukaisen argumentin. Vastauspuussa opiskelijan vastausta verrataan muuttujaan `TAns1`, joka on Maxima-syntaksin muodossa määrittelemämme lauseke.

```
1  <script src="https://moodle.uef.fi/filter/jsxgraph/jsxgraphcore.js">
       </script>
2  <p>
3  Kirjoita ohessa näkyvää funktion kuvaajaa vastaava lauseke käyttäen
4  muuttujaa \( x \). Siis vain lauseke, esimerkiksi<br>
5  <tt>2*sin(x)*x^5</tt>
6  </p>
7  <div class="jxgbox" id={#boardname#} style="width: 400px; height:400px;
       overflow: hidden; position: relative; "> </div>
8
9  <p>Vastaus: [[input:ans1]]</p><div>[[validation:ans1]]</div>
10
11 <script>
12   var board = JXG.JSXGraph.initBoard({#boardname#}, {boundingbox:{#
         boardsize#}, keepaspectratio:false, axis:true, showCopyright:
         false, showNavigation: false, grid: false});
13   board.create('functiongraph', [function(x){return eval({#fx_js#});}
         ],{withLabel:true, name:'\\(y = f(x)\\)'} );
14 </script>
```

Listaus 15: JSXGraph-kuvio kysymyksessä -esimerkin Kysymysteksti-kentän teksti

5.3 STACK-vastauskenttien DOM-manipulointi

Oletetaan seuraavaksi, että olemme määritelleet kysymystekstiin vastauskentän komennolla `[[input:ex1_ans1]]`. STACK muodostaa tästä valmiiseen STACK-kysymykseen seuraavanlaisen HTML-elementin:

```
<input type="text"name="q736708:1_ex1_ans1"id="q736708:1_ex1_ans1"
size="11"style="width: 9.1em"value= />
```

Järjestelmä asettaa määrittelemämme vastauskentän lopulliselle kysymyssivulle `<input>`-tagilla. Vastauskentän tyyppi on tässä algebrallinen lauseke, mikä muodostaa HTML-elementin tyyppiä `text`, ja jonka koko (`size`) ja mahdollinen lähtöarvo (`value`) on määritelty tehtävänasettelussa Syntaksivihje-kentässä. Vastauskentän nimeksi (`name`) ja tunnisteeksi `id` tulee sama arvo, mikä muodostuu aina STACK-järjestelmän generoimasta tunnistearvosta sekä nimeämästämme vastauskentästä. Esimerkissämme vastauskentän tunnistearvo on `q736708:1_` ja tämän perään STACK asettaa syöttämämme vastauskentän nimen eli `ex1_ans1`.

Vastauskenttä voidaan itse syöttämämme vastauskentän nimen ja DOM-manipuloinnin avulla hakea JavaScript-muuttujaksi, jolloin voimme lukea tai syöttää arvoja siihen suoraan JavaScript-ohjelmoinnilla. Arvojen automaattinen syöttäminen vastauskenttään on tarpeen esimerkiksi sellaisissa tehtävissä, joissa haluamme vastauksen olevan jokin opiskelijan asettama JSXGraph-kuvion tila. Tietojen lukeminen vastauskentästä on hyödyllistä esimerkiksi siinä tapauksessa, että tahdotaan tietää onko opiskelija jo vastannut kysymykseen, toisin sanoen onko vastauskentän arvo muu kuin tyhjä merkkijono. Vastauskenttä kannattaa nimetä yksilöllisesti tehtävissä, joissa sitä halutaan muokata. Perustelu tälle selviää hieman myöhemmin tässä Luvussa.

Kun vastauskenttä on nimetty yksilöllisesti, voidaan Kysymysteksti-kenttään kirjoittaa algoritmi Listauksen 16 mukaisesti. Rivillä 1 on suoritettu vastauskentän yksilöllinen nimeäminen. Rivillä 5 luemme DOM:ia hyödyntäen muuttujan `ex1_inputs` tyypiksi listan objekteja, jotka ovat kysymyssivun kaikki `<input>`-elementit.

```
1  <div> [[input:ex1_ans1]] [[validation:ex1_ans1]] </div>
2
3  <script>
4    var ex1_ansfield;
5    var ex1_inputs = document.getElementsByTagName('input');
6
7    // haetaan vastauskenttä
8    for (var i=0; i < ex1_inputs.length; i++) {
9      if (ex1_inputs[i].id.indexOf('ex1_ans1') >= 0)
10       ex1_ansfield = ex1_inputs[i];
11   }
12
13   if (ex1_ansfield.value.length > 0) {
14     ex1_startingpoints = eval(ex1_ansfield.value);
15   }
16   ...
```

Listaus 16: Vastauskentän hakeminen JavaScript-muuttujaksi -algoritmi

Tämän jälkeen riveillä 8-11 asetamme muuttujan `ex1_ansfield` arvoksi sen HTML-elementin, jonka `id` sisältää merkkijonon `'ex1_ans1'`. Se on siis kyseisen tehtävän vastauskenttä,

joka on nyt asetettu JavaScript-muuttujan `ex1_ansfield` arvoksi. Voisimme myös hakea vastauskentän sen nimen perusteella, eli rivi 9 voisi olla myös seuraavanlainen ja koodi toimisi samalla tavalla:

```
if (ex1_inputs[i].name.indexOf('ex1_ans1') >= 0)
```

Listauksessa 17 on selvyyden vuoksi kirjoitettu rivit 8-11 pseudokoodilla.

```
8  käy läpi kaikki listan ex_inputs elementit
9    jos elementin ex_inputs id:stä löytyy merkkijono 'ex1_ans1'
10     aseta ex_ansfield arvoksi kyseinen ex_inputs elementti;
```

Listaus 17: Listauksen 16 rivit 8-11 pseudokoodina

Vastauskentän DOM-manipulointia käytettäessä kannattaa siis vastauskentän `[[input:ans1]]` nimi (kuten täytyy myös validoinnin nimi `[[validation:ans1]]`) muuttaa tehtäväkohtaisesti yksilölliseksi. Tämä siksi, että jos samalla kysymys- tai tenttisivullamme on kaksi tai useampi tehtävä, joissa käytetään vastauskentän muokkausta JavaScript-koodista käsin, voimme hakea eri tehtävien vastauskentät oikeaa käsittelyä varten. Mikäli nimeäisimme kaikkien tehtävien vastauskentän oletusarvoisesti `ans1` ja hakisimme Listauksen 16 tavoin elementin JavaScript-muuttujaksi, mitä tapahtuisi? Elementin tunnisteeseen vertaaminen rivillä 9 muuttuisi

```
if (ex1_inputs[i].id.indexOf('ans1') >= 0)
```

muotoiseksi. Tällöin, koska sivulla on useampi elementti, jonka tunnisteesta löytyy merkkijono 'ans1', jokainen JavaScript-etsintäalgoritmi tallentaisi muuttujan arvoksi viimeisimmän löytämänsä `input`-elementin. Näin ollen vain sivun viimeinen tehtävä toimisi halutulla tavalla.

Vastauskentän tiedon lukeminen ja siihen kirjoittaminen suoritetaan haetun elementin arvoa (`value`) lukemalla tai siihen syöttämällä. Oletetaan, että olemme hakeneet tehtävän vastauskentän muuttujan `ex_ansfield` arvoksi. Tämän kentän sisältämä arvo saadaan lukemalla tieto `ex_ansfield.value`. Jos vastauskenttä on tyhjä,

```
ex1_ansfield.value.length
```

antaa arvon nolla.

Vastauskenttä voidaan hakea käytettäessä muitakin vastauksen tyyppejä kuin algebrallista lauseketta. Jos vastaustyyppi on esimerkiksi matriisi, voidaan tiedot siihen syötetyistä arvoista hakea Listauksen 18 mukaisella JavaScript-koodilla. Matriisin vastauskenttien arvojen hakeminen poikkeaa hieman aiemmin esitetystä tekstikentän hakemisesta. Matriisin elementeille järjestelmä ei luo tunnistetietoja (`id`) ollenkaan, mutta voimme hakea vastauskentät niiden nimen (`name`) perusteella. Matriisin vastauskenttä nimeää matriisin alkiot `[[input:ans1]]`-määrittelyssä luomamme muuttujannimen ja kunkin alkion sijainnin mukaan. Sijaintitiedon järjestelmä asettaa niin, että vasemmassa yläkulmassa oleva alkio on nimetty `ans1_sub_0_0` ja oikean alakulman alkio `ans1_sub_1_1`, kun kyseessä on 2×2-matriisi ja `ans1` on vastauskentäksi määrittelemämme nimi. Muutoin haun periaate on hyvin samankaltainen: käydään läpi kaikki `<input>` elementit ja nimen perusteella tallennetaan tieto muuttujaan.

```
var ansmatrix = [[0,0],[0,0]];
var inputel = document.getElementsByTagName('input');

for (var i=0; i<inputel.length; i++) {
  if (inputel[i].hasAttribute('name')) {
    if (inputel[i].getAttribute('name').indexOf('ans1_sub_0_0')>0)
        ansmatrix[0][0] = inputel[i].value;
    else if (inputel[i].getAttribute('name').indexOf('ans1_sub_0_1')>0)
        ansmatrix[0][1] = inputel[i].value;
    else if (inputel[i].getAttribute('name').indexOf('ans1_sub_1_0')>0)
        ansmatrix[1][0] = inputel[i].value;
    else if (inputel[i].getAttribute('name').indexOf('ans1_sub_1_1')>0)
        ansmatrix[1][1] = inputel[i].value;
  }
}
```

Listaus 18: Matriisin syöttötietojen manipulointi

Mikäli nyt tehtävään ei olla vielä vastattu, saa lista `ansmatrix` arvoikseen tyhjän merkkijonon, muutoin poimitaan opiskelijan syöttämä vastaus palautesivulta. Näin ollen voidaan esimerkiksi tuoda palautesivulla esiin jokin haluttu komponentti tehtävään tai kuvaan, joka on tehtävässä. Listauksessa 19 on esimerkki siitä, kuinka tällainen on tehty eräässä tehtävässä kurssille Lineaarialgebra.

```
if (ansmatrix[0][0].length > 0 &&
ansmatrix[0][1].length > 0 &&
ansmatrix[1][0].length > 0 &&
ansmatrix[1][1].length > 0) {
extraPoint.setAttribute({visible:true});
}
```

Listaus 19: Matriisin syöttötietojen tarkistus ja toiminto

5.4 STACK-JSXGraph -kuvamuotoinen vastaus

Kun halutaan käyttää JSXGraph-kuviota vastauksena STACK-kysymyksessä tarvitsemme keinon, jolla voidaan tulkita onko opiskelijan vastaus oikein. Käytetään tässä yhteydessä tällaisesta vastausmuodosta nimitystä *kuvamuotoinen vastaus*. Tehtävänä voisi olla esimerkiksi jonkin kuvan pisteen asettaminen haluttuun paikkaan, josta opiskelijan painettua Lukitse vastaus -painiketta arvioitaisiin STACK-järjestelmällä onko vastaus oikein.

Tällaisen tehtävätyypin luominen onnistuu STACK-JSXGraph -yhdistelmällä seuraavasti:

- Suoritetaan luvun 5.3 tapaan vastauskentän lukeminen JavaScript-muuttujaksi.
- Luodaan jo(i)llekin JSXGraph-kuvion elementeille tapahtumankäsittelijä(t).
- Tapahtumankäsittelijän funktio syöttää kuviosta haetun arvon (jolle on voitu tehdä laskutoimituksia) vastauskentän arvoksi.
- Opiskelijan painaessa Lukitse vastaus -painiketta suoritetaan tehtävän tarkastus STACK-järjestelmän solmussa.

JSXGraph-elementin tapahtumankäsittelijä suorittaa siis funktion, jonka avulla voidaan tehdä toimintoja kuten millä tahansa JavaScript-kielen funktioilla. Tapahtumankäsittelijä voidaan

määritellä mille tahansa JSXGraph-kuvion objektille. Objektille luodaan tapahtumankäsittelijä suorittamalla sille operaatio `on`, ja asettamalla sille haluttu *tapahtuma* ja toiminto (funktio).

```
Objekti.on('tapahtuma', function(){ toiminto }).
```

Oletetaan edelleen, että olemme hakeneet JavaScript-muuttujalle `ex1_ansfield` arvoksi kysymyksen vastauskenttä-elementin. Listauksessa 20 rivillä 4 on määritelty tapahtumankäsittelijä pisteelle, jonka muuttujannimi on `p1`. Tapahtuma syötetään JSXGraph-objekteille merkkijonona, tässä tapauksessa `'up'`. Tapahtuma `up` toteutuu, kun käyttäjä nostaa hiiren painikkeen (tai sormen kosketusnäytöltä) ylös, jolloin toteutetaan funktion sisällä oleva toiminto. Toiminto voi olla esimerkiksi tässä esitetty pisteen koordinaattien kirjoittaminen vastauskentän arvoksi. Pisteen koordinaatit saadaan haettua JSXGraph-piste -elementin funktioilla `X()` ja `Y()`, jotka palauttavat pisteen x- ja y-koordinaatit vastaavasti. Vastauskentän syöte täytyy olla merkkijono, ikään kuin käyttäjä olisi syöttänyt vastauksen käsin. Tästä syystä syötteeksi kootaan merkkijono, jossa yhdistetään hakasulut, syötettävät alkiot ja alkiot erottava pilkku.

```
1 <script>
2   var board = JXG.JSXGraph.initBoard('Taso', {boundingbox:[-4,4,4,-4]} )
      ;
3   var p1 = board.create('point', [1,1], {name:'A'});
4   p1.on('up', function() {ex1_ansfield.value = '[' + p1.X() + ',' + p1.Y
      () + ']';});
5 </script>
```

Listaus 20: STACK-kuvamuotoinen vastaus: tapahtumankäsittelijä pisteelle `p1`

Muita hyödyllisiä dynaamisissa JSXGraph-kuvioissa käytettäviä tapahtumankäsittelyfunktioita ovat esimerkiksi 'move' (suorittaa tapahtuman aina kun elementtiä siirretään) ja 'down' (suorittaa tapahtuman aina kun elementtiä klikataan tai kosketetaan sormella) (JSXGraph Wiki, 2012). Tapahtumankäsittelijällä voidaan viedä erityyppisiä arvoja syötteeksi vastauskenttään. Edellisessä esimerkissä asetimme vastauskenttään pisteen koordinaatit listana. Tarkastellaan vielä esimerkkiä, jossa viemme vastauskentän syötteeksi janan pituuden. Olkoon edelleen vastauskenttä-elementti haettu muuttujalle `ex1_ansfield`. Rakennamme janan muodostamalla kuvioon ensin kaksi pistettä, ja määrittelemällä sitten janan näiden kahden pisteen välille (Listauksen 21 rivit 3-5). Koska opiskelija voi muuttaa janan pituutta joko pistettä `p1` tai pistettä `p2` siirtämällä, täytyy tapahtumankäsittelijä asettaa molemmille pisteille (Listauksen 21 rivit 6 ja 7). Funktio `L()` palauttaa janan pituuden.

```
1 <script>
2   var board = JXG.JSXGraph.initBoard('Taso', {boundingbox:[-4,4,4,-4]} )
      ;
3   var p1 = board.create('point', [1,1], {name:'A'});
4   var p2 = board.create('point', [-1,-1], {name:'B'});
5   var s1 = board.create('segment', [p1,p2], {strokeWidth:1});
6   p1.on('up', function() {ex1_ansfield.value = s1.L();});
7   p2.on('up', function() {ex1_ansfield.value = s1.L();});
8 </script>
```

Listaus 21: Tapahtumankäsittelijä janalle `s1`

On siis tärkeää asettaa tapahtumankäsittelija jokaiselle sellaiselle elementille, jonka muuttami-

nen vaikuttaa kuviosta kysyttyyn arvoon.

Kuvamuotoisissa vastauksissa ei yleensä haluta, että vastauskenttä `[[input:ans1]]` tai validointi `[[validation:ans1]]` näkyy opiskelijoille. Elementit voidaan piilottaa asettamalla ne Kysymysteksti-kentässä `<div>`- tai `<p>`-tagien väliin, ja muotoilemalla sitten tämän elementin näkyvyyttä CSS-muotoilulla seuraavasti:

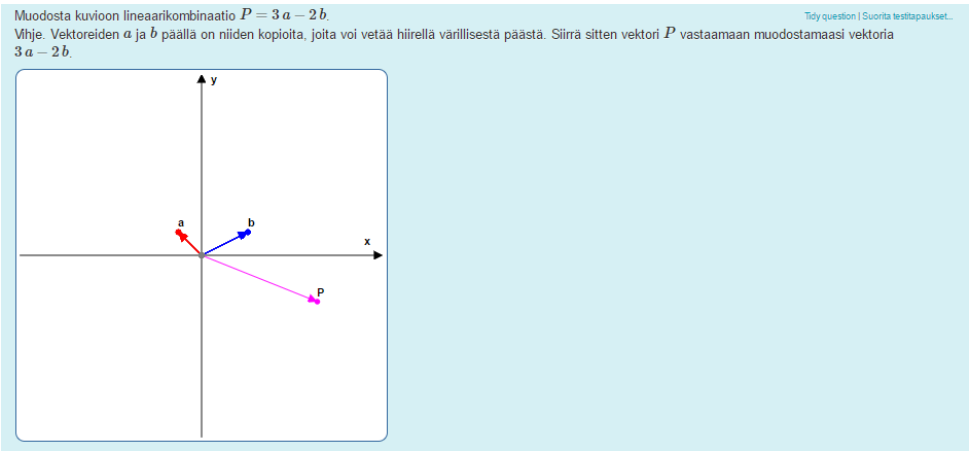
```
<div style='display:none;'>[[input:ans1]] [[validation:ans1]]</div>.
```

Tämän lisäksi on suositeltavaa asettaa Vastaus:ans1-asetuksista sekä Esikatselu- että Näytä validointi -valinnan arvoksi 'Ei', jottei STACK yritä muodostaa vastauksen muodon esikatselua aina kun tapahtumankäsittelijä asettaa uuden arvon vastauskenttään. Usein on tarpeen myös vaihtaa Liukuluvut kielletään -asetuksen arvoksi 'Ei', jos syöttötieto voi olla liukuluku.

Kun opiskelija kuviota muutettuaan painaa Lukitse vastaus -painiketta, siirtyy vastauskenttään asetettu arvo käsiteltäväksi vastauspuuhun, jossa opiskelijan vastauksen arviointia voidaan suorittaa normaalisti. Hyödyllistä on myös se, että opiskelijan vastattua kysymykseen voimme asettaa kuvion siihen tilaan, johon opiskelija sen vastauksessaan asetti. Voimme koodata kysymystekstiin algoritmin, joka poimii vastauskentän sisältämän arvon, ja sen perusteella määrittelee kuvion lähtötilanteen samaksi kuin opiskelija sen vastauksessaan asetti. Tästä on esimerkki Luvussa 5.4.1.

5.4.1 Esimerkkikysymys: kuvamuotoinen vastaus

Tässä käydään läpi erään kurssilla Lineaarialgebra (2017) käytetyn kysymyksen laatimisen vaiheet. Kysymystä käytettiin tentissä, jolla pyrittiin arvioimaan opiskelijoiden lähtötasoa. Kuvassa 23 on kysymyksen esikatselusivu, missä näkyy tehtävänanto ja kysymyksessä käytetty kuvio. Kuvioon on määritelty päällekkäin monta kappaletta siirrettäviä kantavektoreita a ja b, joiden avulla opiskelijan tulee muodostaa tehtävänannon määrittelemä lineaarikombinaatio. Opiskelijan on siirrettävä vektori P siten, että se esittää vaadittua lineaarikombinaatiota, ja tämän jälkeen painettava Lukitse vastaus -painiketta. Tehtävä ei siis sisällä vastauskenttää lainkaan. Kysytty lineaarikombinaatio on satunnaistettu. Listauksessa 22 on Tehtävän muuttujat -kentän määrittelyt.



Kuva 23: Kysymys kurssilta Lineaarialgebra 2017 esimerkissä kuvamuotoinen vastaus

```
1 P_alku: [5,-2];
2
3 a_kerroin: rand([-3,-2,-1,2,3]);
4 /* rajoitetaan hieman b:n kerrointa, jotta pysyy kuvassa*/
5 b_kerroin: if a_kerroin = 3 or a_kerroin = 2 then rand([3,-2]) else if
      a_kerroin = -3 or a_kerroin = -2 then rand([2,-3,-4]) else rand
      ([-4,-3,-2,2,3]);
6 /*kysyttävä vektori */
7 vect: a_kerroin * a + b_kerroin * b;
8
9 TAns1: [a_kerroin * -1 + b_kerroin * 2, a_kerroin * 1 + b_kerroin * 1];
```

Listaus 22: Tehtävän muuttujat -kentän määrittelyt esimerkissä kuvamuotoinen vastaus

Satunnaistaminen on tehty asettamalla lineaarikombinaation kertoimet satunnaisiksi. Vektorin b kerrointa on rivillä 5 rajoitettu ehtolausein, jotta kysytyt lineaarikombinaatiot pysyisivät kuva-alueen sisällä. Kysymyksen mallivastaukseksi asetetaan lasketun lineaarikombinaation koordinaatit.

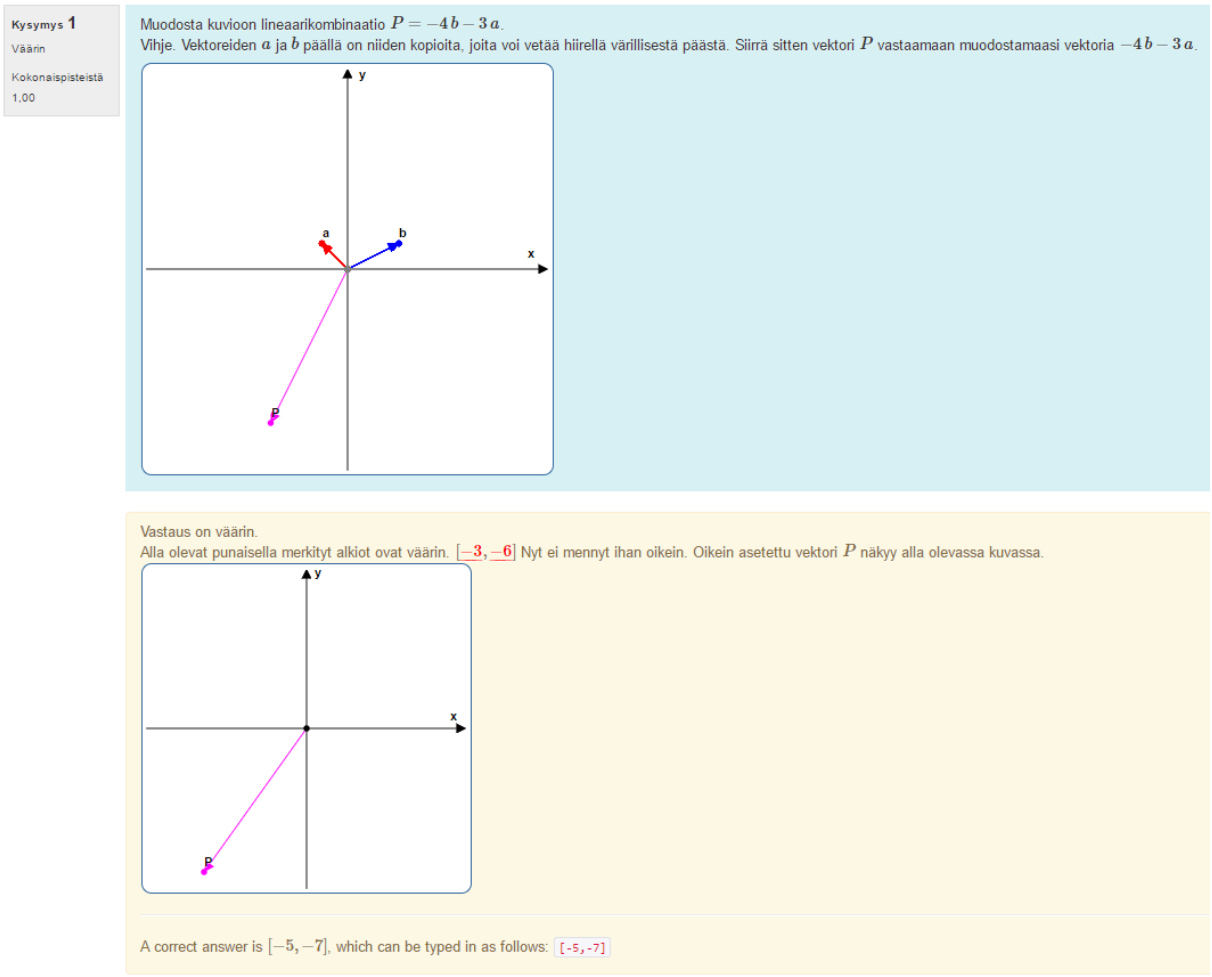
Kysymysteksti-kentän määrittely on Listauksessa 23. Listauksessa 23 näytetään koodin tiivistämiseksi vain pisteen `sgtuotto1New_el[12]` (joka vastaa kuviossa vektorin P loppupistettä) määrittely JSXGraph-koodissa rivillä 32. Vastauskenttä piilotetaan opiskelijalta rivillä 5, ja sen nimi muutetaan. Vastauskenttä haetaan muuttujan `inputSG1new` arvoksi rivien 17-19 algoritmilla. Lisäksi kysymyksessä säilytetään opiskelijan asettaman vektorin P asema kuviossa, kun opiskelija on vastannut ja tarkastelee vastauksesta saamaansa arviointia. Tätä varten luodaan muuttuja `ansCoordSG01new` (rivillä 21), joka sisältää vektorin P koordinaatit riippuen siitä, onko kysymykseen vastattu. Muuttujalle asetetaan arvo `{#P_alku#}`, joka on määritelty Tehtävän muuttujat -kentässä. Algoritmi riveillä 23-26 tarkistaa onko vastauskentän arvon pituus suurempi kuin nolla (eli onko opiskelija vastannut kysymykseen), ja asettaa muuttujan `ansCoordSG01new` arvoksi vastauskentän sisältämän arvon. Mikäli vastauskenttä on tyhjä, asetetaan siihen vektorin P lähtöarvo, jottei opiskelija voi vastata tyhjää (liikuttamatta vektoria P lainkaan). Rivillä 32 luodaan kuvioon vektori P ja asetetaan sen koordinaatiksi muodostamamme lähtöarvo.

Kysymyksessä on asetettu jokainen kantavektorin ja vektorin P kärkipiste JSXGraph-optiolla `snapToGrid` liikkumaan ainoastaan kuvion määrittelemällä ruudukolla (grid). Tämä auttaa paitsi opiskelijaa ratkaisemaan tehtävän, myös itse tehtävän laadintaa. Kun asetetut vektorit ovat aina täsmällisesti koordinaatiston määrittelemillä kokonaisilla koordinaateilla, ei vastausta arvioidessa ole tarpeen tarkistaa onko se jonkin virhemarginaalin sisällä.

Rivillä 37 määritellään kuvion vektorille P tapahtumankäsittelijä, joka hiiren painikkeen nostolla asettaa vastauskenttään sen loppupisteen koordinaatit. Vastauskentän määrittelyissä on asetettu Liukuluvut kielletään-, Esikatselu- ja Näytä validointi -arvoksi 'Ei'. Vastauksen arviointi onnistuu tässä suoraviivaisesti, kun voimme verrata vastauskentän arvoa suoraan Tehtävän muuttujat -kentässä määriteltyyn `TAns1`-arvoon. SAns-kenttään siis tulee vastauskentästä luettava muuttujan arvo `sg1_new_ans1`, ja TAns1-kenttään muuttuja `TAns1`. Lisäksi 'Solmun 1 palaute jos väärin' -kentässä piirretään kysymyksen väärän ratkaisun palautteeksi oikeaan asemaan sijoitettu vektori P. Tätä varten luodaan toinen (pienemmäksi skaalattu) kuvio muutoin samalla JSXGraph-koodilla kuin kysymystekstissäkin, mutta muuttujien nimet on vaihdettu, kantavektoreita a ja b ei näytetä, ja vektorin P loppupisteeksi asetetaan `{#TAns1#}`. Kuvassa 24 näkymä väärän vastauksen jälkeen.

```
<p>Muodosta kuvioon lineaarikombinaatio \( P =  {@vect@}\).
<br>
Vihje. Vektoreiden \( a \) ja \( b \) päällä on niiden kopioita, joita voi vetää hiirellä värillisestä päästä. Siirrä sitten vektori \( P \) vastaamaan muodostamaasi vektoria {@vect@}.</p>

<p style="display:none;">[[input:sg1_new_ans1]] [[validation:sg1_new_ans1]]</p>

<p>
<script src='https://moodle.uef.fi/filter/jsxgraph/jsxgraphcore.js'></script>

<div class='jxgbox' id='SGtuotto1New' style='width: 400px; height: 400px; overflow: hidden; position: relative; background-color:white;'></div>
<script>//![CDATA[
(function(){

var inputsSG01new = document.getElementsByTagName('input');
  var inputSG01new;

  for (var i=0; i<inputsSG01new.length; i++) {
    if (inputsSG01new[i].id.indexOf('sg1_new_ans1') >= 0)
      inputSG01new = inputsSG01new[i]; }

  var ansCoordSG01new =  {#P_alku#};

  if (inputSG01new.value.length>0) {
     ansCoordSG01new = eval(inputSG01new.value);
  }
  else inputSG01new.value = "[" + ansCoordSG01new + "]";

var SGtuotto1New_b = JXG.JSXGraph.initBoard('SGtuotto1New', {
    boundingbox:[-8,8,8,-8], originX:8, originY:8, zoomX:1, zoomY:1,
    keepaspectratio:false, axis:false, showCopyright: false,
    showNavigation: false, grid: false} );var sgtuotto1new_el=[],
    sgtuotto1new_transf=[], sgtuotto1new_transf2=[],
    sgtuotto1new_animOn=false;

...

sgtuotto1new_el[12] = SGtuotto1New_b.create('point', ansCoordSG01new, {
    name:'P',label:{offset:[0,10]},color:'magenta',size:1,
    showInfoBox:false,snapToGrid:true});

...

/*event handler for 'P' */
  sgtuotto1new_el[12].on('mouseup', function() {
     inputSG01new.value = '['+sgtuotto1new_el[12].X()+','+
        sgtuotto1new_el[12].Y()+']';
  });
})();//]]>
</script>
</p>
```

Listaus 23: Lineaarikombinaatio-kysymyksen Kysymysteksti-kenttä



Kuva 24: Lineaarikombinaatio-kysymyksen väärän vastauksen palaute

5.5 Muita huomioita kysymysten konstruoinnista

Kun algoritmilla viedään syöte kuviosta vastauskenttään tapahtumankäsittelijän avulla, voidaan vastaukseksi syötettävää arvoa muokata JavaScript-koodin avulla (kuten jana pituus Luvussa 5.4). Joskus se voi olla suoraviivaisempi tapa tarkistaa onko opiskelijan asettama kuvamuotoinen vastaus oikea. JavaScript ja siihen saatavat ohjelmakirjastot (kuten JSXGraph) sisältävät monipuolisempia funktioita kuin Maxima. JSXGraph-ohjelmakirjasto sisältää paljon geometristen elementtien ominaisuuksien laskentaan räätälöityjä funktioita. Esimerkiksi JSXGraph-elementin piste (point) eräs metodi on Dist(), jonka avulla pisteen etäisyyttä toiseen pisteeseen voi kätevästi verrata.

```
piste1.on('up', function() {
     if (piste1.Dist(piste2) < 0.5) vastauskentta.value = 'true';
     else vastauskentta.value = 'false';
});
```

Listaus 24: JSXGraph-pisteen metodin Dist() käyttö

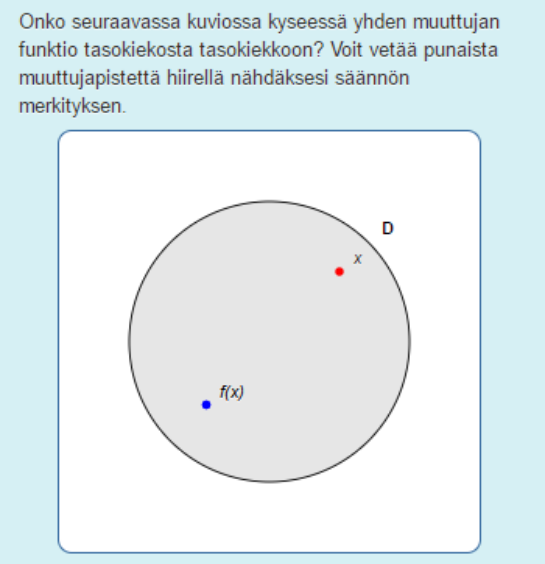
Listauksessa 24 vastauskenttään asetetaan merkkijono "true", mikäli objektien piste1 ja piste2 etäisyys toisistaan on vähemmän kuin 0,5, muutoin "false". Vastauspuussa ei näin ollen tarvitse suorittaa laskutoimituksia, vaan riittää verrata syötettä TAns1-kenttään asetetulla true-arvolla. Tämä on vain yksi esimerkki JavaScriptin mahdollisuuksista.

Joskus dynaamisiin kuvioihin on hankalaa määritellä ominaisuuksia, joita haluaisimme kysymyksessä visualisoida. Esimerkiksi jonkin objektin katoaminen tai ilmestyminen kuviossa jonkin tietyn säännön mukaan voi olla vaikeaa. JSXGraph-koodilla voidaan luoda hyvinkin monipuolisesti dynaamisuutta, jota muutoin olisi hankala toteuttaa. Tarkastellaan tätä tarkemmin erään Lineaarialgebra-kurssilla käytetyn kysymyksen rakennetta läpi käyden.

```
1  <script>
2
3  var board = JXG.JSXGraph.initBoard('board', {boundingbox:[-3,3,3,-3],
       keepaspectratio:false, axis:false, showCopyright: false,
       showNavigation: false} );
4  var circ = board.create('circle', [[0,0], 2], {name:'<b>D</b>',
       withLabel:true, strokeColor:'black', strokeWidth:1, fillColor:'grey'
       , fillOpacity:0.2, fixed: true} );
5  var p1 = board.create('point', [1,1], {name:'<em>x</em>', color:'red',
       size:1, showInfobox:false} );
6
7  p1.on('drag', function () {
8      //point stays in circle
9      if (p1.Dist(p0) > circ.Radius()) {
10         projectionCoordinates = JXG.Math.Geometry.projectPointToCircle(
               p1, circ);
11         p1.moveTo(projectionCoordinates.usrCoords);
12     }
13 } );
14 var p2 = board.create('point', [function(){return -0.9*p1.X();},
       function(){return -0.9*p1.Y();}], {name:'<em>f(x)</em>', color:'blue
       ', size:1, showInfobox:false} );
15 var p3 = board.create('point', [function(){return p2.X();}, function(){
       if (p1.Y()>=0) return p2.Y(); else return p2.Y()*0.4}], {name:'<em>f
       (x)</em>', color:'blue', size:1, showInfobox:false} );
16 </script>
```

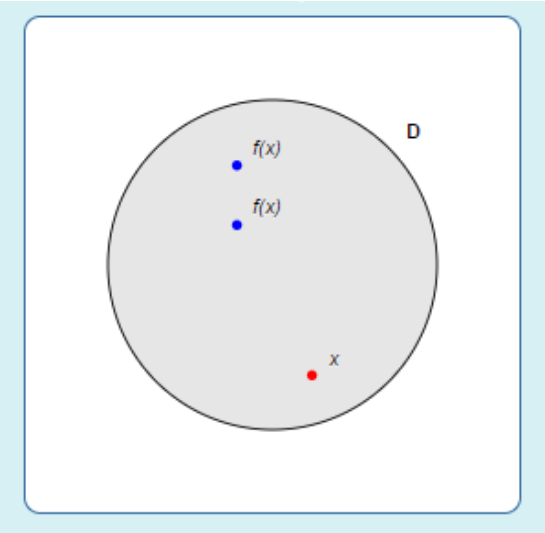
Listaus 25: JSXGraph-pisteen määrittely funktion avulla

Kysymyksessä täytyy kuviota tutkimalla ja kokeilemalla tutkia, määrääkö kuvion sääntö funktion tasokiekosta tasokiekkoon. Kuviossa voi siirtää pistettä x, jolloin funktion kuvapiste $f(x)$ liikkuu kuviossa. Tässä kysymyksessä haluttiin, että tietyssä tasokiekon alueessa x:llä on kaksi kuvapistettä. Tämä on monella dynaamisen geometrian sovelluksella olla hankala toteuttaa, mutta JSXGraph-koodissa voimme määritellä kuvion elementtien ominaisuuksia funktioilla, jotka voivat sisältää ehtolauseita. Listauksessa 25 on kysymyksen kuvion muodostava JSXGraph-koodi.



Kuva 25: Lineaarialgebra-kurssin Funktiot-harjoittelussa käytetty JSXGraph-kuvio

Kuvioon luodaan kiekko $\mathbf{D}$ ja piste x riveillä 4 ja 5. Rivien 7-13 algoritmilla tapahtumanseuraajaa hyödyntämällä saadaan piste x pysymään kiekon sisällä. Rivillä 14 muodostetaan pisteen x kuvapiste asettamalla sen koordinaatit riippumaan pisteen x koordinaateista. Kuvapisteen x-koordinaatti on pisteen x koordinaatti kerrottuna luvulla $-0{,}9$, ja y-koordinaatti vastaavasti. Kuvapisteen päälle luodaan toinen kuvapiste, jonka koordinaateiksi luetaan äsken määrittelemämme kuvapisteen koordinaatit sillä ehdolla, että kun pisteen x y-koordinaatti on negatiivinen tämän kuvapisteen y-koordinaatti on toisen kuvapisteen y-koordinaatti kerrottuna luvulla $0{,}4$. Kuvio 26 esittää tilannetta, jossa piste x on siirretty kiekon alaosaan eli kuviossa y-akselin negatiiviselle puolelle.



Kuva 26: Lineaarialgebra-kurssin Funktiot-harjoittelussa käytetty JSXGraph-kuvio, jossa pistettä x on siirretty

6 Työkokeilun analysointi ja yhteenveto

Tässä kappaleessa tarkastelen Moodle-kysymysten laadinnan onnistumista seuraavista näkökulmista: STACK-tehtävien edut ja rajoitteet, ajankäyttö, järjestelmien toimivuus sekä pedagogiset edut ja haasteet. Myös opiskelijoilta kerättyä palautetta tehtävistä ja näiden tehtävien kautta oppimisesta käsittelen tässä kappaleessa. Tutkielman yhteenveto on Luvussa 6.3.

6.1 Työn vaiheiden arviointi

Järjestelmän hallinta ja hyvä osaaminen vie jonkin verran aikaa, mutta suhteellisen yksinkertaisten kysymysten teko ei vaadi käyttäjältä kovin suurta osaamista. Käytännössä paras tapa oppia järjestelmä on kokeilla ja testata tehtävien tekoa. Vaikkei itselläni ollut ennen työn aloittamista käsitystä STACK-järjestelmän ominaisuuksista ja mahdollisuuksista, jo ensimmäisellä kokeilukerrallani sain rakennettua yksinkertaisen kysymyksen pelkästään tarkastelemalla muutamia valmiiksi ABACUS-projektiin tuotettuja kysymyksiä.

Järjestelmän oppiminen on toki yksilöllistä. Mutta vaikka kysymysten teko onnistuisikin osaamisen puolesta, niiden rakentamiseen kuluu aikaa kohtalaisesti. Riippuen kysymyksen rakenteen monimutkaisuudesta, yhden kysymyksen laadintaan kului aikaa tunnista muutamiin päiviin. Työn aikana kokeiltiin useita uusia tehtävärakenteita, joista näkyvimpänä olivat JSXGraph-kuvioita sisältävät ja niiden rakentamiseen kului enemmän aikaa. Lisäksi ongelmat kysymysten ja niiden elementtien näkymisessä vaivasivat usein, mikä vei taas enemmän aikaa.

Vaikka STACK-kysymysten tekoon kuluukin enemmän aikaa kuin perinteisten paperille laadittujen kysymysten tekoon, on sillä kuitenkin huomattavia etuja. Kysymyksistä ei tarvitse tehdä useita eri versioita, kun järjestelmässä on mahdollista satunnaistaa kysymysten arvoja. Lisäksi valmiista kysymyksistä on pienellä vaivalla muokattavissa uusia kysymyksiä tekemällä (vain pieniä) muutoksia tehtävän muuttujiin, laskutoimituksiin tai kysymystekstiin. STACK-tehtävien selvä etu on myös se, että ne säilyttävät tiedon opiskelija-arvioinnista järjestelmässä, sisältäen arvioinnin niin yksittäisissä kysymyksissä onnistumisesta kuin arvioinnin kurssin kokonaissuoriutumisesta.

JavaSketchpadToJSXgraph-kääntäjän ohjelmoinnista muodostui iso osa työnkuvaa. Ohjelman suunnittelu oli melko suoraviivainen prosessi, eikä vienyt paljon aikaa. Kääntäjän ensimmäisen version ohjelmointiin kului aikaa ainoastaan noin viikko, mutta siinä oli vain pieni osa JavaSketchpad-kielen konstruktioista. Ensimmäisessä versiossa toimivat ainoastaan konstruktiot piste, suora, jana, ympyrä ja teksti sekä muutamia translaatioita kuten esimerkiksi JSP-komento `VectorTranslation`. Käytännössä kääntäjän ohjelmointi eteni sitä mukaa kun käännettävissä JSP-kuvioissa tuli vastaan uusia, kääntäjälle tuntemattomia komentoja. Viimeisin versio kääntäjästä sisältää kaikki JSP-komennot, mutta aivan kaikki JSP-koodit eivät kääntäjällä muunnu JSXGraph-muotoon ilman pientä JSP-koodin muokkausta ennen käännöstä. Kääntäjä säästi huomattavan määrän aikaa dynaamisten kuvioiden rakentamisessa.

6.2 Havaintoja kysymysten laadinnasta

Kun ensimmäisiä kysymyksiä koostettiin tenttiympäristöön havaittiin, että JSXGraph-kuviot eivät aina näkyneet tentin tulosten tarkastelusivulla. Syyksi osoittautuivat samannimiset muuttujat JSXGraph-koodeissa. On siis huomioitava, että näissä muuttujat nimetään yksilöllisesti kuhunkin JSXGraph-kuvion sisältämään kysymykseen, mikäli niitä käytetään samalla Moodle-sivulla (tai yleisemmin HTML-sivulla). Kysymyksen (yksilöllisen) nimen mukainen muuttujien nimeäminen todettiin toimivaksi ratkaisuksi, jolloin ongelma ei ilmene kysymysten jatkokäytössä. STACK-muuttujat sen sijaan saavat olla samannimisiä eri tehtävien välillä.

STACK-tehtävien laatimiseen havaitsin ainoaksi vaihtoehdoksi Pelkkä teksti -tyyppisen editorin käytön, sillä muilla editoreilla sekä LaTeX-muotoiluissa että JavaScript-koodin kirjoituksessa esiintyi ongelmia. Esimerkiksi jotkin JavaScript-käskyt merkitsevät TinyMCE-editorissa myös jotakin käskyä. Tärkeää on myös huomioida, että JavaScript- ja JSXGraph-koodia kirjoitettaessa Moodlen tekstieditorin on oltava HTML-muodossa, jotta kysymysteksti menee kysymykseen puhtaassa HTML-muodossa eikä tekstieditorin muotoilemana (Kuva 14).

Usein kysymystä laadittaessa kysymysteksti muodostui hyvinkin pitkäksi. Kysymystekstin kirjoittaminen Moodle-tekstieditoreilla osoittautui silloin hankalaksi ja virheiden havaitseminen JSXGraph-koodista oli vaikeaa. Ratkaisu oli käyttää jotakin toista, koodin kirjoittamiseen sopivampaa tekstieditoria. Saman havainnon teki Muli (2014) tutkielmassaan. STACK-tehtäviä laatiessa työlästä oli keskeneräisten kysymyksen esikatselu, mikä onnistui vasta kysymyksen tallentamisen jälkeen. Kysymyksen tallentaminen "nollasi"aina kysymyksenlaadintanäkymän, jolloin jatkaminen siitä mihin ennen esikatselua jäi oli hidasta. Samoin esikatselutyökaluun olisi toivonut jonkin option, jolla kysymyksessä muodostetut ja lasketut muuttujien arvot olisivat näkyneet tehtävien kehittäjälle. Jouduin ratkaisemaan tämän ongelman syöttämällä muuttujien arvoja kysymystekstikenttään, jotta myös loppukäyttäjälle näkymättömiä muuttujien arvoja pystyi tarkastelemaan. Tämäkin siis osaltaan hidasti tehtävien laadintaa. Tehtävän muuttujien muokkausvaiheen tulostamiseen on kuitenkin tiettävästi tulossa parannus syksyllä 2017.

Ongelmia ilmeni LaTeX-tekstin kanssa. STACK-muuttujiksi määritellyt joukot eivät näkyneet kysymyksissä oikein. Esimerkiksi STACK-muuttujissa määritelty joukko

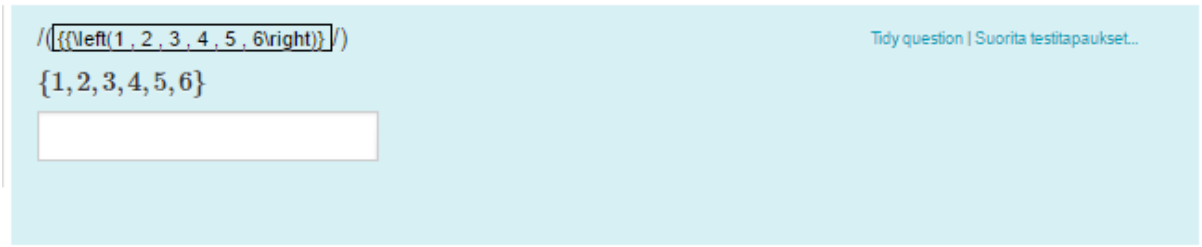
```
joukko: {1,2,3,4,5,6};
```

ei näkynyt missään CASText-kentässä oikein kun sitä kutsuttiin komennolla `{@joukko@}`. Tämä haittasi paljon tehtävien tekoa, eikä ongelma selvinnyt kunnolla missään vaiheessa. Totesimme myös, että erään toisen yliopiston Moodle-ympäristössä tätä ongelmaa ei ollut. Tämä ongelma voitiin kiertää esimerkiksi kirjoittamalla CASText-kenttiin joukko syntaksilla

```
\( \{ {#joukko#}\} \).
```
(4)

Esiintynyt näkyvyysongelma on Kuvassa 27 ylempänä ja alempana on joukon tulostus koodin 4 tavalla.

LaTeX-ongelmia esiintyi myös JSXGraph-kuvioiden objektien nimien kanssa. Kuvioita rakentaessamme nimesimme usein kuvion objekteja LaTeX-muodossa, sillä se on mahdollista ja toimikin aluksi hyvin. Maaliskuussa 2017 LaTeX-muotoilu kuvioiden nimissä kuitenkin lakkasi toimimasta, eikä ongelman syy ratkennut tämän tutkielman valmistumiseen mennessä. Kun LaTeX-muotoilu toimi, se tapahtui kirjoittamalla JSXGraph-elementin nimi muodossa



Kuva 27: LaTeX-ongelma joukkojen tulostuksessa

```
board.create('point', [1,1], { name: '\\(x\\)' } );
```

eli aivan kuten normaalissa LaTeX-syntaksissa, mutta kahdella kenoviivalla `\\`. Näin kirjoitettuna JavaScript-kieli tulostaa kuvioon yhden kenoviivan ja sulun LaTeX-muotoon käännettävän tekstin molemmille puolille.

Kun kuvioita käännettiin JSP-muodosta JSXGraph-muotoon jouduttiin alkuperäistä JSP-koodia hieman muuttamaan joko JSXGraph-kuvion ulkoasun eroavaisuuden takia tai jos alkuperäistä kuviota haluttiin muutoin korjata tai muuttaa. Totesimme hyväksi ratkaisuksi sisällyttää kysymystekstiin tehtävässä esitettävän JSXGraph-koodin lisäksi myös alkuperäinen JSP-koodi kommenttina. Tämä nopeutti tehtävien muokkaamista, mikäli käännetyssä kuviossa havaittiin jokin virhe tai asia joka haluttiin muuttaa.

Kysymyksissä käytetyt dynaamiset JSXGraph-kuviot toimivat pääasiassa hyvin. Eräissä kuvioissa oli kuitenkin toimivuuteen liittyviä ongelmia. Esimerkiksi kurssilla Lineaarialgebra (2017) käytetty kuvio (kuva 23) toimi virheellisesti useilla opiskelijoilla. Kuvio sisälsi useita päällekkäisiä kantavektoreita a ja b, joita opiskelija pystyi siirtämään ja asettelemaan kuvioon. Moni opiskelija kohtasi vektoreita siirtäessään virheen, jossa hiiren osoittimella siirretty vektori ei irronnutkaan osoittimesta kun hiiren painikkeen nosti ylös. Vektori ikäänkuin liimautui hiiren osoittimeen. Tätä ongelmaa yritettiin selvittää, mutta testitilanteissa ongelmaa ei onnistuttu itse toistamaan. Myöhemmin kuviosta tehtiin myös paranneltu versio (uudemmalla kääntäjän versiolla), ja opiskelijoita pyydettiin kokeilemaan sekä alkuperäisen että uuden kuvion toimivuutta. Kuudestatoista opiskelijasta yhdellä esiintyi edellä mainittu ongelma kuvion vanhemman version kanssa, mutta uudempi versio kuviosta toimi. Muilla opiskelijoilla ongelmaa ei esiintynyt kummankaan version kanssa. Oletettavaa siis on, että vika johtui jostakin virheestä vanhan version (käännetyssä) koodissa, mutta tarkka vian syy jäi selviämättä.

6.3 Opiskelijapalaute

Opiskelijoilta kerättiin palautetta STACK-järjestelmän ja -kysymysten toimivuudesta ja hyödyllisyydestä. Palautetta kerättiin Moodle-tenttien yhteydessä kursseilla Matematiikan johdantokurssi 2016 (Funktiotesti) sekä Lineaarialgebra 2017 (Lineaarialgebran alkutesti ja Funktiokoe). Lisäksi kesäkuussa 2017 tehtiin kysely, jossa palautetta kysyttiin yleisesti näiden kahden kurssin aikana käytetyistä STACK-kysymyksistä.

Tähän on poimittu eräitä kursseilta saatuja avoimia kommentteja edellämainittuihin asioihin liittyen. Opiskelijapalautteen analysoiminen kursseilla kerätyistä palautteista on haastavaa, koska

on vaikea tietää johtuivatko tekniset ongelmat kysymyksistä, laitteistosta vai käyttäjästä itsestään. Palautetta kurssien aikana kertyi kuitenkin runsaasti, ja eräät huomiot palautteissa toistuvat. Käsittelen tässä vain näitä yleisimmin toistuneita, olellisesti samaan asiaan liittyviä huomioita ja kommentteja kysymyksistä.

Selvästi eniten ongelmia kysymyksissä aiheuttivat dynaamiset kuviot, joiden näkyvyydessä, toimivuudessa ja selkeydessä oli hankaluuksia. Suurimmalla osalla kuviot toimivat, ja ne saivat kiitosta, mutta hyvin joillakin kuviot eivät näkyneet lainkaan, tai niissä esiintyi kuvion elementteihin liittyviä toimivuusongelmia. Eräässä tentissä arviolta puolella opiskelijoista ilmeni ongelmia kuvioiden kanssa. Ongelmat eivät vaikuttaisi johtuneen verkkoselaimesta, sillä ne esiintyivät useassa selaimessa. Muutamia pomittuja kommentteja liittyen dynaamisten kuvioiden ongelmiin:

"Jostain syystä dynaamiset kuviot eivät tänään toimineet minulla chromessa, mutta eilen toimivat."

"...kun tartuin kuvan ympyrän pisteeseen. Kun ympyrän pisteestä otti otteen, ei se vapautunut vaikka hiiren painikkeen nosti, vaan piti saada jotenkin muuten deaktivoitua..."

"Älykkäät kuvat ei toimineen ollenkaan apuvektorit jäivät hiireen kiinni eivätkä irronneet samoin myös P vektori."

Myös syntaksin syöttämiseen liittyviä haasteita ilmeni, ja osalla oli vaikeuksia löytää tarvittavat merkit (kuten sulut) tietokoneen näppäimistöltä. Selvästi on nähtävissä, että tämän tyyppisiin kysymyksiin vastaaminen on osalle opiskelijoista uutta, ja että ympäristöön tutustuminen ennen varsinaisiin arvioitaviin kysymyksiin siirtymistä on tarpeen. Opettajan on välttämätöntä osata ja ymmärtää käytettävää tekniikkaa. Tehtävänannoissa pyrittiin olemaan täsmällisiä ja selkeitä, ja niistä opiskelijat antoivatkin hyvää palautetta. Tämä osaltaan auttoi syntaksin kirjoittamisessa, kun tehtävänannossa oli kerrottu selkeästi missä muodossa vastaus täytyy järjestelmään syöttää. Selkeät toimintaohjeet siis vaikuttavat kaventavan oppilaiden teknisen osaamisen välisiä tasoeroja. Muutama kommentti liittyen vastausten kirjoittamiseen:

"Vastauksen oikea kirjoitusohje kaikkiin tehtäviin, jossa voi tulla vaaraksi vastata väärällä tavalla (aalto- vai hakasulkeet, pillku vai piste)."

"Sulkujen oikein laittaminen oli hieman vaikeaa!"

Kysymysten antaman palautteen on syytä olla mahdollisimman yksityiskohtaista. STACK-järjestelmällä suoritettu opiskelijan tietokoneavusteinen arviointi toimii parhaiten, jos kysymyksiin saadaan rakennettua monipuolista ja informatiivista palautetta. Palautteen mukaan voisi olla syytä myös harkita, että tentti antaisi selkeämmin palautetta kunkin tehtävän jälkeen, eikä vasta siinä vaiheessa kun kaikkiin tehtäviin on vastattu. Muutama kommentti järjestelmän antamasta palautteesta:

"Tentistä saisi vielä enemmän irti, jos vaihtoehdon kohdalla kävisi ilmi syy sille, miksi se on väärä vaihtoehto."

"Olisi hyvä, jos väärästä vastauksesta saisi palautteen samantein, että mikä meni vikaan ja miksi."

Moni opiskelija koki Moodle-kysymykset hyödyllisiksi ja mielekkäiksi tehdä. Dynaamiset ku-

viot auttoivat joidenkin opiskelijoiden kohdalla matemaattisen käsitteen ymmärtämistä. Opiskelijat kokivat, että Moodle-tentit voivat parhaimmillaan toimia kertausmateriaalina varsinaista tenttiä varten. Kurssipalautteissa kommentoitiin asiaa esimerkiksi seuraavasti:

"Näiden funktiotestien ja harjoitusten jälkeen alan pikkuhiljaa ymmärtää jotain funktioista."

"Onhan tämä hyvä tarkistuspiste, siitä mitä pitäisi osata. Jotkut asiat eivat ekalla kerralla meinanneet aueta."

"Tämä harjoitus avasi käsityksen lineaarialgebrasta ja auttoi muistelemaan vektorilaskentaa"

Kurssien jälkeen teetetystä kyselystä saatiin jossain määrin saman kaltaisia kommentteja. Kyselyyn vastasi ainoastaan 16 opiskelijaa, joten kovin tarkkaa analyysia sen tulosten perusteella ei voida tehdä. Kyselyn tulokset kuitenkin vahvistavat edellä mainittuja kursseilla kerättyjä palautteita. Kyselyssä opiskelijoilta selvitettiin kuinka he kokivat seuraavat asiat kysymyksiin liittyen:

- STACK-Maxima -syntaksin mukainen kirjoittaminen vastaukseksi,
- STACK-järjestelmän tuottaman automaattisen palautteen, ja kysymyksiin kirjoitetun palautteen hyödyllisyys,
- kuinka merkittävänä he pitivät dynaamisten kuvioiden tuomaa lisäarvoa matemaattisten käsitteiden oppimisessa,
- oliko tehtävänanto riittävän selkeä, varsinkin kysymyksissä, joissa kuvio toimi vastauksena.

Opiskelijoilta kysyttiin, kuinka hankalana he pitivät järjestelmän vaatiman syntaksin kirjoittamista STACK-kysymysten vastauskenttään. Noin 63% vastanneista ei pitänyt syntaksin kirjoittamista lainkaan vaikeana, ja noin 37% vain osittain vaikeana. Hankalana syntaksin kirjoittamista ei pitänyt kukaan. Avoimet kommentit olivat hyvin samankaltaisia kuiin kurssien aikana saadussa palautteessa:

"Kun tajusi merkitsemistavan, oli helppo. Mutta silti mielestäni kuitenkin ihan typerä tapa."

"Aluksi emme saaneet näppäimistöstä toimimaan sulkeita. Ilman teknisiä ongelmia ei enää ollut vaikeuksia."

Toiseksi kysyttiin järjestelmän antaman palautteen hyödyllisyyttä. Väitteestä, että järjestelmä antoi riittävästi palautetta opiskelijan vastauksista, noin 31% oli täysin samaa mieltä, 50% osittain samaa mieltä ja 19% osittain eri mieltä. Lisäksi kysyttiin, oliko järjestelmän antama palaute oli hyödyllistä oppimisen kannalta. Noin 44% piti palautetta hyödyllisenä, 31% osittain hyödyllisenä, 19% vain vähän hyödyllisenä ja yksi vastaaja ei lainkaan hyödyllisenä. Kysymykset siis jossain määrin kaipasivat kattavampaa palauteosion määrittelyä. Usein tehtäviä laatiessa palauteosio jäi turhan pienelle huomiolle, kun varsinaisen kysymyksen rakenteen laatimiseen kului paljon aikaa. Usein myös täsmällisen palautteen laatiminen tehtäviin oli haasteellista. Järjestelmän antamaa palautetta kommentoitiin kyselyssä seuraavasti:

"Annetut palautteet olivat hyviä, mutta joissakin tapauksissa palaute oli vain "väärin"tai "oikein". Toisaalta kyse taitaa olla vain kuvausten kirjoittamisesta."

"Kun jokaisesta tehtävästä sai heti palautteen, oli helppo oppia virheistään."

"Palaute ei aina perustellut oikeaa vastausta."

Kolmanneksi kysyttiin, kuinka dynaamiset kuviot auttoivat matemaattisten käsitteiden ymmärtämisessä. Noin 87% oli täysin tai osittain samaa mieltä.

"Hyvä visualisointi, erityisesti linalgissa, auttaa ymmärtämään eri konsepteja. Lisää plussaa visualisoinneista, joissa näkyi myös matriisit ym (esim. miltä eri determinantit "näyttää")"

"Auttavat kyllä, mutta oikean vastauksen päättely oli välillä hankalaa, sillä kuviota ei aina saanut pysäytettyä haluamaansa kohtaan, ja sen liike oli liian nopeaa tarkemmalle havainnoimmiselle."

Kaksi vastaajaa oli kuitenkin osittain tai täysin eri mieltä. Tähän saattoi kuitenkin vaikuttaa se, että kuvioiden toimivuuden kanssa oli usein ongelmia, mikä varmasti osittain turhautti kysymyksiin vastaamista. Lisäksi tehtävänanto täytyy olla selkeä ja täsmällinen.

"Dynaamiset kuviot ymmärsi silloin kun ne oli selitetty hyvin. Välillä selitykset dynaamiselle kuviolle oli sekavia."

"Erittäin hyödyllisiä harjoitteita. Auttoivat hyvin paljon asioiden ymmärtämisessä."

Neljännellä kysymyksellä selvitettiin nimenomaan tehtävänantojen selkeyttä kysymyksissä, joissa vastaus oli kuvamuotoinen. Opiskelijoilta kysyttiin oliko kysymysten tehtävänanto selkeä ja helposti ymmärrettävä. Noin 37% vastaajista oli tätä mieltä, 50% oli osittain samaa mieltä ja noin 12% osittain eri mieltä. Kysymystyyppi oli opiskelijoille varmasti uusi, kuten myös tapa jolla vastaus syötettiin. Näin ollen tehtävänantoon tällaisissa kysymyksissä täytyy kiinnittää erityistä huomiota, jotta opiskelijalle on selvää kuinka kuviota voi (ja täytyy) muuttaa.

"Välillä jouduin miettimään tehtävänantoa liian paljon. Ts. tehtävänanto oli sekavaa."

"Mielestäni dynaamiset tehtävät ovat hyödyllisiä. On vain tärkeää, että kaikki toimii sujuvasti. Koetilanteessa monella tuli teknisiä ongelmia ja niitä piti sitten selvitell?. Lisäksi useilla opiskelijoilla taisi olla vaikeuksia ymmärtää tehtävänantoja, varsinkin kun vastaus piti olla tietyssä muodossa annettuna."

Lisäksi opiskelijoita pyydettiin asettamaan (tärkeys)järjestykseen seuraavat kysymystyypit siinä mielessä, kuinka tärkeänä he oppimisen kannalta kyseistä kysymystyyppiä pitävät:

- Kysymykset, joissa tulos lasketaan ja syötetään matemaattisena lausekkeena annetun mallin mukaan.
- Kysymykset, jotka liittyvät dynaamiseen kuvioon ja vastaus syötetään matemaattisena lausekkeena annetun mallin mukaan.
- Kysymykset, jotka liittyvät dynaamiseen kuvioon ja vastaus annetaan kuviota säätämällä.
- Kysymykset, joissa kysytään teoria-asioita ja vastaus annetaan yksi- tai monivalintana.
- Kysymykset, joissa voi laskemalla saada tuloksen ja antaa vastaus yksi- tai monivalintana.
- Kysymykset, joihin vastataan sanallisesti.

Kuvassa 28 on kyselyn tulokset. Tuloksesta nähdään, että opiskelijat selvästi kokivat hyötyvänsä dynaamisia kuvioita sisältävistä tehtävistä. Avoimissa kommenteissa kävi ilmi, että tällaisten tehtävien ratkaiseminen oli jossain määrin mukavampaa kuin muiden tehtävätyyppien.

	1. tärkein	2. tärkein	3. tärkein	4. tärkein	5. tärkein	6. tärkein
Kysymykset, joissa tulos lasketaan ja syötetään matemaattisena lausekkeena annetun mallin mukaan.	6,25 %	25,00 %	37,50 %	12,50 %	12,50 %	12,50 %
Kysymykset, jotka liittyvät dynaamiseen kuvioon ja vastaus syötetään matemaattisena lausekkeena annetun mallin mukaan.	43,75 %	18,75 %	6,25 %	12,50 %	6,25 %	6,25 %
Kysymykset, jotka liittyvät dynaamiseen kuvioon ja vastaus annetaan kuviota säätämällä.	25,00 %	37,50 %	25,00 %	6,25 %	6,25 %	0,00 %
Kysymykset, joissa kysytään teoria-asioita ja vastaus annetaan yksi- tai monivalintana.	18,75 %	6,25 %	6,25 %	25,00 %	56,25 %	0,00 %
Kysymykset, joissa voi laskemalla saada tuloksen ja antaa vastaus yksi- tai monivalintana.	6,25 %	0,00 %	25,00 %	31,25 %	18,75 %	6,25 %
Kysymykset, joihin vastataan sanallisesti.	0,00 %	12,50 %	0,00 %	12,50 %	0,00 %	75,00 %

Kuva 28: Opiskelijoiden mielipide kysymystyyppien tärkeydestä oppimisen kannalta kursseilla Matematiikan johdantokurssi (2016) ja Lineaarialgebra (2017)

Myös kysymyksiä, joissa vastaus annettiin matemaattisena lausekkeena, pidettiin tärkeinä. Suhtautumisessa tähään kysymystyyppiin oli kuitenkin hajontaa, ja vain yksi vastaaja koki tämän kysymystyypin itselleen kaikkein tärkeinpänä. Syntaksin syöttämiseen liittyvät haasteet ja välillä hankalat tehtävänannot saattavat osittain selittää tätä tulosta, koska tällaiset matemaattiset kysymykset olivat monelle opiskelijalle vieraita.

6.4 Yhteenveto

Moodle-ympäristöön voidaan STACK-järjestelmällä laatia hyvinkin monipuolisia kysymyksiä. STACK ja JSXGraph toimivat pääosin hyvin keskenään, ja niillä tuotetut tehtävät toimivat eräitä poikkeuksia lukuunottamatta moitteetta. Muutamissa kysymyksissä ilmenneiden teknisten ongelmien syyt jäivät valitettavasti epäselväksi. STACK-JSXGraph -kysymykset, joiden ratkaisu oli kuvamuotoinen olivat teknisesti haastavin osa työtä. Loppuvaiheessa ne saatiin toimimaan luotettavasti. JSXGraph-kuvioita sisältävät STACK-kysymykset ovat vain yksi esimerkki siitä joustavuudesta, jonka STACK-jäjestelmä tarjoaa.

Kysymysten laadinta tällä järjestelmällä on melko helppoa, mikäli tarkoituksena on tehdä suoraviivaisia ja rakenteeltaan yksinkertaisia kysymyksiä. Mikäli kysymyksistä haluaa monipuolisempia esimerkiksi vastauksen arvioinnin osalta, joutuu järjestelmän ominaisuuksiin tutustumaan tarkemmin. Maxima-syntaksin tunteminen on hyödyllistä ja verkosta löytyvät ohjeet ja dokumentaatiot auttavat Maximan komentojen ja funktioiden käytössä. JSXGraph-kuvioita sisältävien kysymysten laadintaa helpottaa JavaScript-kielen alkeiden osaaminen, mutta enempää ei tarvita. Myös kuvamuotoinen vastaus -kysymysten laadinta onnistuu jo vähäisellä ymmär-

ryksellä JavaScript-ohjelmoinnista. Toivottavasti tässä tutkielmassa esitetyt ohjeet ja esimerkit auttavat kysymysten laadinnassa.

Dynaamisten kuvioiden rakentaminen nopeutui huomattavasti JavaScriptToJSXGraph-kääntäjän myötä. Kääntäjä kehittyi työjakson aikana paljon ja se osoittautui loppuvaiheessa toiminnaltaan hyvinkin luotettavaksi. Alkuperäiset JavaSketchpad-kuviot kääntyvät rakenteeltaan hyvin vastaaviksi JSXGraph-kuvioiksi. Kääntäjästä oli apua työjakson aikana usein pelkkänä JSXGraph-koodin tuottajana; JavaSketchpad-syntaksi on monessa mielessä helpompaa (ja selkeämpääkin) kirjoittaa kuin JSXGraph-syntaksi, joten sillä on hyvä rakentaa JSXGraph-kuvioita.

Järjestelmän automaattinen arviointi toimii parhaimmillaan hyvin, ja sillä on mahdollista tuottaa hyvin informatiivista ja oppimista edistävää palautetta. Arviointi toimii luotettavasti, kunhan kysymyksen laadinnassa ei ole tullut virheitä tai huomioon ottamattomia seikkoja. Vastauspuiden luominen ja linkittäminen on tehty helpoksi. Palautteen ja tehtävien kirjoittaminen on selkeää, ja CASText-muotoilun ansiosta matemaattisten ilmaisujen kirjoittaminen ja näyttäminen tuo kysymyksiin lisäarvoa. Samoin lisäarvoa tuo mahdollisuus satunnaistaa kysymyksien muuttujia.

Vaikka tietokoneita ja teknologiaa on nykyään lähestulkoon kaikkialla ympärillämme, ei tekniikan osaamista voida pitää itsestään selvänä. Joillakin opiskelijoilla on vaikeuksia tämänkaltaisten kysymysten ratkaisemisessa, ja usein ne johtuvat enemmän teknisestä osaamisesta kuin matemaattisen käsitteen ymmärtämisestä. Edelleen on siis tärkeää, että käytettäessä tietotekniikkaa matematiikan opetuksen työvälineenä, opiskelijoita perehdytetään tekniikan käyttöön. Tietokoneharjoitusten teettäminen Moodle-ympäristössä ennen painoarvoltaan tärkeämpää tenttiä, on hyvä keino. Erityisen tärkeää on, että opettaja itse hallitsee opetuksessa tarvittavan tekniikan käytön. Parhaimmillaan tällaiset tietokoneharjoitteet ovat mainio apuväline opiskelijoiden osaamistason testaamiseen ja asioiden kertaamiseen. Tätä arviota tukee myös opiskelijapalaute.

Dynaamiset kuviot matemaattisen käsitteen ymmärtämisessä ovat oikein sovellettuna tehokkaita apuvälineitä. Tärkeää on pedagogisten puitteiden asettaminen; mitä halutaan saavuttaa dynaamista geometriaa sisältävillä kuvioilla? Opiskelijat kokivat työjakson aikana laaditut kuviot mielekkäiksi ja oppimista edistäviksi, kunhan niiden toimivuus teknisesti voidaan taata. Kursseilla Lineaarialgebra ja Matematiikan johdantokurssi käytetyt dynaamiset kuviot olivat opiskelijoiden mielestä tärkeitä apuvälineitä matemaattisten käsitteiden oppimisessa. Ajoittaiset tekniset ongelmat eivät näyttäisi vaikuttaneen tähän suhtautumiseen negatiivisesti; vaikka joillain opiskelijoilla ilmeni teknisiä ongelmia, pitivät he silti dynaamisten kuvioiden merkitystä käsitteen oppimisessa tärkeänä.

Viitteet

Bindner, D. ja Martin, E. (2011). *A Student's Guide to The Study, Practice, And Tools of Modern Mathematics*. Chapman & Hall/CRC, Taylor & Francis Group.

Borovik, A. (2017). Information technology in university-level mathematics teaching and learning: a mathematician's point of view. *Research in Learning Technology*, 19:1:73–85. Luettu osoitteesta: `http://dx.doi.org/10.1080/09687769.2010.548504`.

Burn, B., Appleby, J., ja Maher, P. (1998). *Teaching Undergraduate Mathematics*. Imperial College Press.

Crockford, D. (2008). *JavaScript: The Good Parts*. O'Reilly Media Inc. Kirja luettu osoitteesta: `http://bdcampbell.net/javascript/book/javascript_the_good_parts.pdf`.

Dautermann, W. (2016). Maxima 5.39.0 manual. Luettu viimeksi 19.5.2017 osoitteesta `http://maxima.sourceforge.net/docs/manual/maxima.html`.

Dubinsky, E. ja Tall, D. (1991). *Advanced Mathematical Thinking and the Computer*. Kluwer Academic Publishers, Netherlands.

Gerhauser, M., Miller, C., Valentin, B., Wassermann, A., ja Wilfahrt, P. (2011). Jsxgraph - dynamic mathematics running on (nearly) every device. *The Electronic Journal of Mathematics and Technology*, 5(1). Luettu osoitteesta: `https://php.radford.edu/~ejmt/deliveryBoy.php?paper=eJMT_v5n1a3`.

Grabmeier, J., Kaltofen, E., ja Weispfenning, V. (2003). *Computer Algebra Handbook: Foundations, Applications, Systems*. Springer.

Haapasalo, L. (2007). Adapting mathematics education to the needs of ict. *The Electronic Journal of Mathematics and Technology*, 1(1). Luettu osoitteesta: `https://php.radford.edu/~ejmt/deliveryBoy.php?paper=eJMT_v1n1p1`.

Heid, M. K. ja Edwards, M. T. (2001). Computer algebra systems: Revolution or retrofit for today's mathematic's classrooms?

Hollebrands, K. F. (2016). Characterizing questions and their focus when pre-service teachers implement dynamic geometry tasks. *The Journal of Mathematical Behavior*, 43:148–164. Luettu osoitteesta `http://doi.org.ezproxy.uef.fi:2048/10.1016/j.jmathb.2016.07.004`.

JSXGraph Wiki (2012). Adding events 2. Internet lähde. Luettu osoitteesta: `http://jsxgraph.uni-bayreuth.de/wiki/index.php/Adding_events_2`.

Kor, L. K. ja Chuah, J. B.-P. (2014). Cognitive workload and mathematics instructional design for non-users of mathematical software applications. *The Electronic Journal of Mathematics and Technology*, 8(4):274–285.

Laborde, C. (2002). *International Journal of Computers for Mathematical Learning*, 6:283 – 317.

Majander, H. (2010). Tietokoneavusteinen arviointi kurssilla diskreetin matematiikan perusteet. Matematiikan ja tilastotieteen laitos, Helsingin yliopisto. Pro gradu -tutkielma.

Majander, H. ja Rasila, A. (2011). Experiences of continuous formative assessment in engineering mathematics. pages 197–214. Tampereen yliopistopaino Oy - Juvenes Print. Silfverberg, H., Joutsenlahti, J. (eds.), Tutkimus suuntaamassa 2010-luvun matemaattisten aineiden opetusta.

Market Share Reports (2010). . Internet lähde. http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0.

MoodleDocs (2016). Moodlen dokumentaatio. Internet lähde. Luettu 7.6.2017 osoitteesta: https://docs.moodle.org.

Moreno-Armella, L., Hegedus, S., ja Kaput, J. (2008). From static to dynamic mathematics: Historical and representational perspectives. *Educational Studies in Mathematics*, (68):99–111.

Muli, P. (2014). Relaatiokäsitteen ymmärtäminen - harjoittelu ja testaus moodle-ympäristössä. *Itä-Suomen yliopisto. Fysiikan ja matematiikan laitos.* Matematiikan pro gradu -tutkielma.

Negrino, T. ja Dori, S. (2007). *JavaScript - Tehokas Hallinta*. Pearson Education.

Nishizawa, H., Yoshioka, T., Pesonen, M. E., ja Viholainen, A. (2012). Interactive worksheets for learning the connection between graphics and symbolic object representation. Proceedings of the 17th Asian Technology Conference in Mathematics (ATCM 2012).

Pesonen, M. E., Haapasalo, L., ja Ehmke, T. (2006). Critical look at dynamic sketches when learning mathematics. *The Teaching of Mathematics*, 9:19–29.

Pesonen, M. E., Haapasalo, L., ja Lehtola, H. (2002). Looking at function concept through interactive animations. The Teaching of Mathematics, pages 37–45.

Rasila, A. (2013). Verkkopohjainen harjoittelu osana matemaattisen ajattelun kehittymisprosessia. pages 26–33. Tampereen yliopisto. Viteli, J., Östman, A. (eds.), Tuovi 11: Interaktiivinen tekniikka koulutuksessa 2013 -konferenssin tutkijatapaamisen artikkelit.

Rasila, A. (2016). E-assessment material bank abacus. In *EDULEARN16 Proceedings*, pages 898–904. IATED. 8th International Conference on Education and New Learning Technologies.

Rasila, A. ja Sangwin, C. J. (2016). Development of stack assessment to underpin mastery learning. Proceedings of the 13th International Congress on Mathematical Education, Hamburg.

Ruthven, K. ja Hennessy, S. (2002). A practitioner model of the use of computer-based tools and resources to sup/port mathematics teaching and learning. pages 47 – 88.

Sangwin, C. (2012). *Computer Aided Assessment of Mathematics*. Springer International Publishing Switzerland.

Sangwin, C. (2015). Computer aided assessment of mathematics using stack. pages 695–713.

Sinclair, N. ja Yurita, V. (2008). To be or to become: how dynamic geometry changes discourse. *Research in Mathematics Education*, 10:2:135–150. Luettu osoitteesta: http://dx.doi.org/10.1080/14794800802233670.

STACK Documentation (2016). Internet lähde. Luettu 2016-10-13 osoitteesta http://stack.bham.ac.uk/moodle/question/type/stack/doc/doc.php/.

STACK Documentation, Answer tests (2016). Internet lähde. Luettu 2017-02-06 osoitteesta http://stack.bham.ac.uk/moodle/question/type/stack/doc/doc.php/Authoring/Answer_tests.md.

STACK Documentation, CAS (2016). Internet lähde. Luettu 2017-04-24 osoitteesta http://stack.bham.ac.uk/moodle/question/type/stack/doc/doc.php/CAS/index.md.

Stylianou, D. A. ja Dubinsky, E. (1999). Determing linearity: the use of visual imagenary in problem solving. pages 245–252. Columbus. Proceedings of the 21th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education.

Sutherland, R. (2006). *Teaching For Learning Mathematics*. McGraw-Hill Education.

The Geometer's Sketchpad Resource Center (2014). The geometer's sketchpad resource center. Internet lähde. Luettu 25.4.2017 osoitteesta: http://www.dynamicgeometry.com/General_Resources/The_Sketchpad_Story.html.

Topic, D. (2016). Moving to a plugin-free web. Kirjoittaja on *Principal Product Managerin* yrityksessä Oracle. Luettu 2.5.2017 osoitteesta https://blogs.oracle.com/java-platform-group/moving-to-a-plugin-free-web.

Viholainen, A. (2008). *Prospective Mathematics Teachers' Informal and Formal Reasoning About the Concepts of Derivative and Differentiability*. University of Jyväskylä.

Vinner, S. (1991). *The Role of Definitions in Teaching And Learning of Mathematics*. Kluwer Academic Publishers, Netherlands.

Wiest, L. R. (2001). The role of computers in mathematics teaching and learning. *Computers in the Schools*, 17(1-2):41–55.

Liite 1: Opiskelijapalautteen kyselylomake

Kysymys 1. Ohessa näet kuvakaappauksen eräästä kysymyksestä kurssilta Lineaarialgebara 2017, jossa vastaus syötettiin matemaattisena ilmaisuna tietokonelaskennan järjestelmän vaatimassa muodossa. Esimerkin tehtävässä vastaus täytyi syöttää joukkona, kyseisessä tapauksessa {1,6}.

Tällaisen järjestelmän vaatiman syntaksin kirjoittaminen ei ollut minulle hankalaa. *

- Täysin samaa mieltä
- Osittain samaa mieltä
- Osittain eri mieltä
- Täysin eri mieltä

Halutessasi voit tarkentaa vastaustasi tähän

Mikä on joukon $\{1,4,9\}$ **kuvajoukko** oheisen taulukon mukaisessa kuvauksessa, kun muuttujana on x?

x	1	2	3	4	5	6	7	8	9	10
y	1	4	9	1	10	6	4	4	6	10

Kysymys 2. STACK-järjestelmän eräs ominaisuus on sen antama automaattinen palaute. Useisiin tehtäviin lisättiin myös tarkempaa palautetta onnistumisesta tai epäonnistumisesta. Mikä on oma kokemuksesi järjestelmän/tehtävien tarjoamasta palautteesta?

Järjestelmä antoi riittävästi palautetta vastauksistani. *

- Täysin samaa mieltä
- Osittain samaa mieltä
- Osittain eri mieltä
- Täysin eri mieltä

Järjestelmän antama palaute oli hyödyllistä oppimisen kannalta. *

- Täysin samaa mieltä
- Osittain samaa mieltä
- Osittain eri mieltä
- Täysin eri mieltä

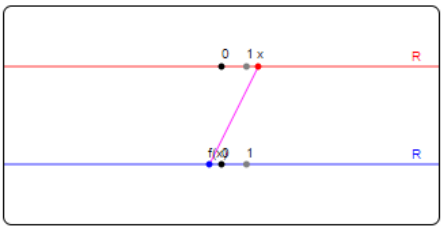
Halutessasi voit tarkentaa vastaustasi tähän

Kysymys 3. Useat tehtävät sisälsivät *dynaamista geometriaa* sisältäviä kuvioita. Dynaamisella geometrialla tarkoitetaan konstruktiota, jossa tietyt ominaisuudet ja säännönmukaisuudet säilyvät, samalla kun jotakin objektia siirretään hiirellä. Ohessa on eräs esimerkkikuvio kurssilta Matematiikan johdantokurssi 2016.

Tällaiset dynaamiset kuviot auttoivat minua hahmottamaan matemaattista käsitettä. *

- Täysin samaa mieltä
- Osittain samaa mieltä
- Osittain eri mieltä
- Täysin eri mieltä

Halutessasi voit tarkentaa vastaustasi tähän



Kuva 29: Liite 1 (1/3): Opiskelijapalautteen kysymykset 1-3

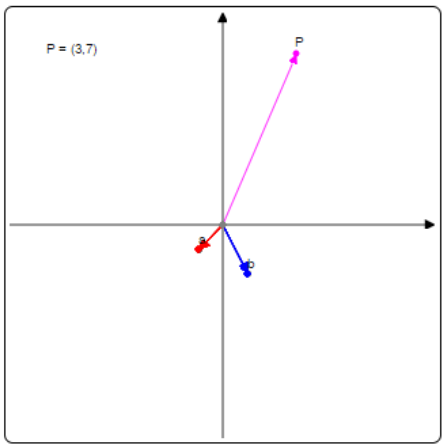
Kysymys 4. Osassa dynaamista geometriaa sisältäviä tehtäviä kokeiltiin uudenlaista vastausmetodia, jossa kuviota täytyi muuttaa, ja muutetun kuvion tila toimi vastauksena. Tämä asettaa uusia haasteita myös tehtävänasetteluun, jotta opiskelija ymmärtää tehtävänannon oikein.

Tehtävänanto tällaisissa tehtävissä oli selkeä ja helposti ymmärrettävä. *

- Täysin samaa mieltä
- Osittain samaa mieltä
- Osittain eri mieltä
- Täysin eri mieltä

Halutessasi voit tarkentaa vastaustasi tähän

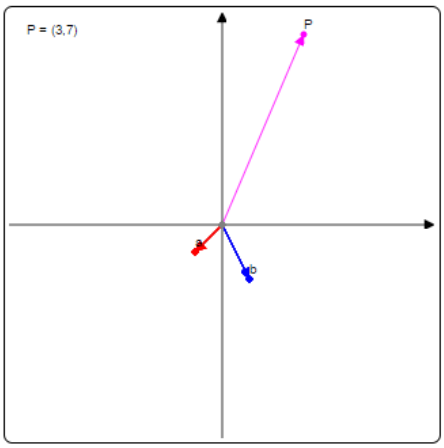
Kysymys 5. Kuviot rakennettiin JSXGraph ohjelmalla. Osissa kuvioista ilmeni ongelmia, joita ei missään vaiheessa saatu selvitettyä. Syy saattoi olla selainversiossa, JSXGraph-ohjelmassa, tai Moodlessa itsessään. Ohessa on eniten ongelmia aiheuttanut kuvio A, jossa monella siirrettävät nuolikuviot jäivät kiinni hiireen. Vertailun vuoksi viereen on konstruoitu sama kuvio uudelleen hieman muunnetulla koodilla, kuvio B. Kokeile, toimivatko kuviot tässä lomakkeella oikein: vektoreista a ja b on useita kopioita, joita pitäisi pystyä siirtelemään hiirellä tarttumalla.



Kuvio A

Toimiko kuvio A oikein? *

- Kyllä
- Ei



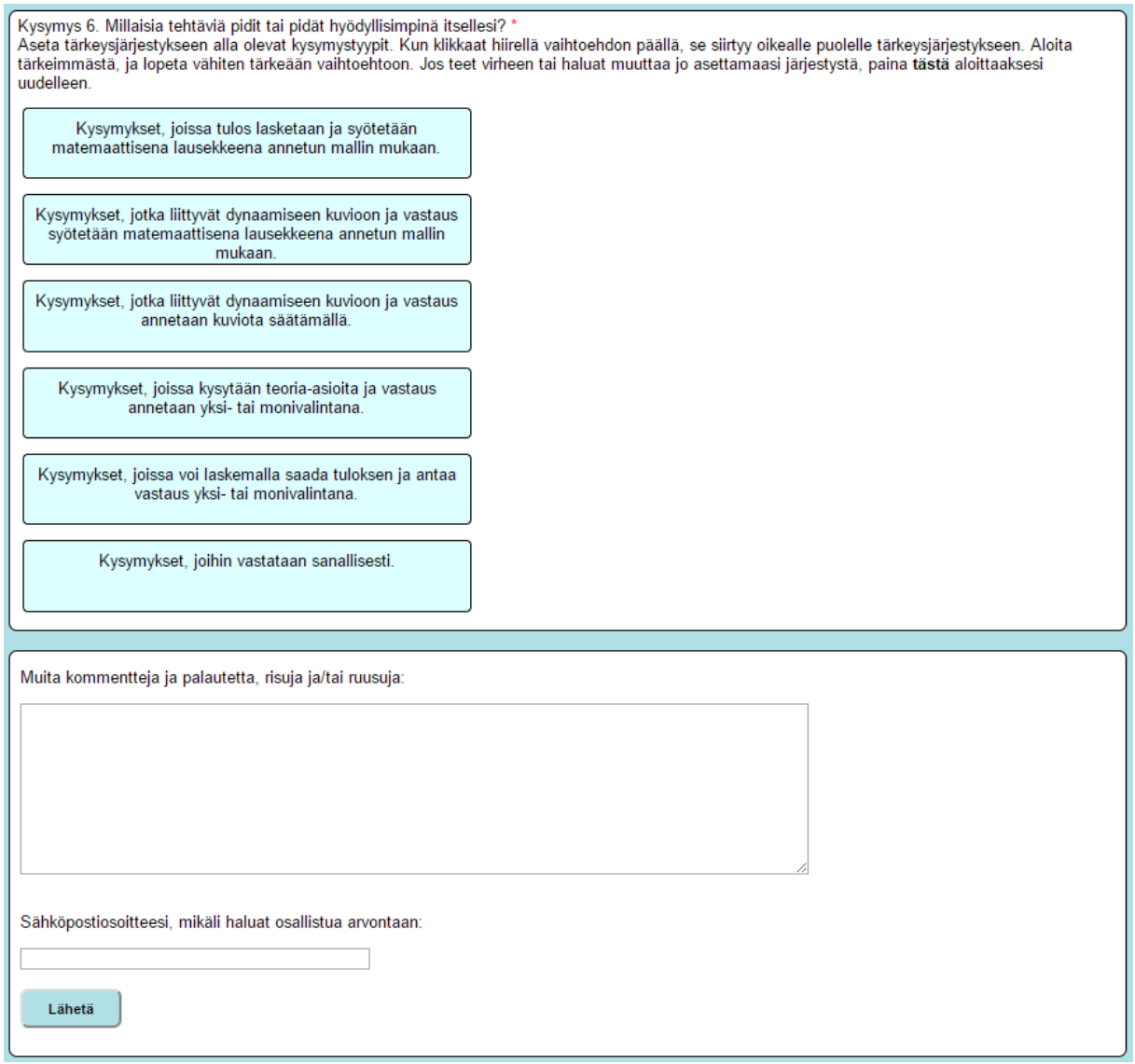
Kuvio B

Toimiko kuvio B oikein? *

- Kyllä
- Ei

Kyselylomake hakee tähän automaattisesti tietoja järjestelmästäsi, jotta voidaan tutkia johtuuko ongelmat jostain tietystä selainversioista. Korjaa nämä tiedot, jos selaintiedot ovat virheelliset tai puuttuvat kokonaan. Selaimesi: Chrome (versionro 58), Käyttöjärjestelmäsi: Windows

Kuva 30: Liite 1 (2/3): Opiskelijapalautteen kysymykset 4-5



Kuva 31: Liite 1 (3/3): Opiskelijapalautteen kysymys 6 ja avoin palaute