jamk.fi

Pelinkehityksen työvälineet opetuskäyttöön

Pelimoottorit, frameworkit ja sovelluskehykset

Jani-Matti Mäkelä

Opinnäytetyö
Marraskuu 2014

Tietojenkäsittelyn koulutusohjelma
Liiketalouden ja hallinnon ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä(t) Mäkelä, Jani-Matti	Julkaisun laji Opinnäytetyö	Päivämäärä 17.11.2014
	Sivumäärä 50	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)
Työn nimi **Pelinkehityksen työvälineet opetuskäyttöön** Pelimoottorit, frameworkit ja sovelluskehykset		
Koulutusohjelma Tietojenkäsittely		
Työn ohjaaja(t) Mika Karhulahti		
Toimeksiantaja(t)		

Tiivistelmä

Opinnäytetyössä tutkitaan toimintatavoiltaan erilaisia pelimoottoreita ja niiden kehitysympäristöjä, pohjautuen Javascript-kieleen, opettajien kanssa tehdyn kartoituksen pohjalta. Tutkimuksen tekijä testasi ja arvioi itse työkalut ja niiden kehitysympäristöt sekä kirjasi niistä havainnot tarpeen mukaan, tutkimusmenetelmänä on siis kvalivatiivinen eli laadullinen menetelmä. Tutkimuksen mittakaavan rajoittaminen oli tärkeää, että pysyttiin aikataulussa, ja lopulta saatiin tarvittavat neljä erilaista työkalua, joiden käyttöön voidaan syventyä tarpeen tullessa.

Tutkimusaineistosta saatiin neljä erilaista pelinkehityksen työkalua, joissa oli joko enemmän tai vähemmän graafinen käyttöliittymä tai sen osa. Jokainen pelimoottori esitellään erikseen. Itse kehittämispuolta ko. pelimoottorilla ei käydä läpi, koska se toteutetaan Javascript-kielellä, jota opetetaan paitsi pelinkehityksessä, sitä hyödynnetään myös mm. sovelluskehityksen opettamisessa, jolloin se tukee myös muita Jyväskylän ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman tavoitteita. Tärkeää oli myös, että jokaisen pelimoottorin kehitysympäristöt ja työkalut olivat saatavilla ilman erillisen lisenssin ostoa, jotta myös itse opiskelijoilla olisi mahdollisuus käyttää opetettavaa pelimoottoria opetuksen ulkopuolella.

Avainsanat (asiasanat)
pelinkehitys; mobiilipelit; HTML5; Javascript; opetustyö; yliopistot; ammattikorkeakoulut; tietojenkäsittely; kvalitatiivinen menetelmä; framework; sovelluskehys; pelimoottori; kehitysympäristö; käyttöliittymä

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
Mäkelä, Jani-Matti	Bachelor´s Thesis	17112014
	Pages 50	Language Finnish
		Permission for web publication (X)

Title
Game development tools for teaching
Game engines, frameworks and application frameworks

Degree Programme
Business Information Systems

Tutor(s)
Karhulahti, Mika

Assigned by

Abstract

This thesis examines the modes of operation of different game engines and development environments, based on the Javascript language teachers on the basis of a survey. The researcher tested and evaluated the game engines and development environment, as well as the findings of the book as needed. The research method was qualitative. Scale limitation of the study was important to stay on schedule, and finally gave the necessary four different game engines, in which usage each individual can concentrate on when needed.

The research material brought four different game engines, which was more or less a graphical user interface or a part of it, and each of the game engine are presented separately. The development process with the game engine isn't discussed, because it is implemented in JavaScript , which is taught not only in the game development, it is also used software development in teaching, in which case it also supports Jyväskylä's University of Applied Sciences of Information processing training program objectives. It was also important that each game engine development environments and tools were available without the purchase of a separate license, in order to students be able to use the game engine to learn outside of teaching.

Keywords
game development; mobile games; HTML5; JavaScript; teaching; universities; community colleges; data processing; qualitative method; framework; application framework; the game engine; development; user interface

Miscellaneous

SISÄLTÖ

KUVAT

KUVIOT

1 Johdanto

Opinnäytetyön opiskelija suorittaa Jyväskylän Ammattikorkeakoululle, tietojenkäsittelyn koulutusohjelmalle ja sen vastaavalle opettajalle. Lisäksi jokaista pelinkehityksestä vastaavaa opettajaa haastatellaan sähköpostitse tai kasvokkain tarpeiden kartoittamiseksi.

Opinnäytetyön tarkoitus on esitellä mahdollisia Open Source-pohjaisia pelimoottoreita, frameworkeja eli sovelluskehyksiä, sekä luokkakirjastoja, joita voidaan hyödyntää Tikon (tietojenkäsittelyn koulutusohjelma) järjestämissä pelinkehitysopinnoissa, joiden perusteena toimii HTML5-tekniikka. Perimmäinen tarkoitus on luoda perusta, josta voidaan siirtyä tarvittaessa aina alustakohtaiseen käsittelyyn asti.

Ottaen huomioon, että Suomen tämänhetkinen pelinkehitys potee työvoimapulaa, voidaan sanoa että tästä on myös opinnäytetyön tekijälle hyötyä, sillä tutkimuksen suorituksen jälkeen tekijällä kuuluisi olla mielessä vähintään perusteet erinäisistä pelimoottoreista, sovelluskehyksistä ja luokkakirjastoista.

2 Tutkimusasetelma

2.1 Opinnäytetyön tavoitteet ja rajaukset

Aiheenvalinnan taustalla oli tekijän kiinnostus peliteollisuudesta ja pelinkehityksestä, lähinnä graafisen puolen käyttö pelinkehityksessä. Pelimoottorilla ei varsinaisesti ohjelmoida, vaan se on korkean tason pelinkehitysväline. Framework:it eli sovelluskehykset ja luokkakirjastot sisältävät ohjelmointia mm. HTML5- ja JavaScript-tekniikalla. Sovelluskehyksiä sisällytetään tutkimukseen

lähinnä sen takia, että ne ovat oikeastaan perustason opetusvälineitä ohjelmointiin, sillä mm. JavaScript-osaamisesta on hyötyä myös web- ja mobiilikehityksessä.

Työkalujen on myös sisällettävä mahdollisuus kosketusnäytöille, resoluutioiden järkevälle ylläpidolle sekä käyttöliittymän suunnitteluun kosketusnäytöille. Työkalujen lopullinen toteutus ei myöskään saisi riippua tietystä tyylistä, vaan olisi mahdollisimman yleismaallinen.

Tutkimuksessa siis tutkitaan sekä pelimoottoreita että sovelluskehyksiä ja luokkakirjastoja, jotta saadaan mahdollisimman monipuolista valikoimaa opettajien opetustarpeiden täyttämiseksi. Näitä työkaluja valitaan neljä.

Kaikki työkalut, joita tutkimuksessa käytetään, ovat Open Sourceen eli avoimeen lähdekoodiin pohjautuvia, mikä tarkoittaa että kyseessä olevia työkaluja tai sen osia voidaan käyttää vapaasti ilman erillisiä kustannuksia. Lisäksi tutkitaan saatavuuden mukaan työkaluja, joista on saatavilla ilmainen, joskin mahdollisesti karsittu, versio, jota voitaisiin käyttää opetustarkoituksiin.

Lisäksi kaikilla työkaluilla keskitytään vain 2D-pelien toteuttamiseen, vaikka työkalu tarjoaisi välineet myös kolmiulotteisten pelien toteutukseen. 3D-kehitystä voidaan tarjota edistyneemmille opiskelijoille, mutta se edellyttää jatkotutkimusta.

Tutkimuksen lopputuloksen tarkoitus on esitellä kuinka valitut työkalut eroavat toisistaan ja mikä niistä olisi paras hyödyntää opetustarkoituksessa. Lisäksi helppokäyttöisyys on tärkeä piirre haetuissa työkaluissa, jotta voidaan suositella parhaiten sopivat työkalut sekä opetus- että opiskelukäyttöön.

Lisäksi vertailussa hyödynnetään osaa niistä kriteereistä, joita käytetään mm. Tikon HTML5 Game Development –opintojaksolla:

- Tuki tarvittaville alustoille, kuten tabletit ja älypuhelimet
- Mitä työkalut sisältävät: joustava käyttöliittymä, integroitu fysiikkamoottori, isometrinen tai 3D-tuki, verkkopelituki jne.
    - Isometrinen projektio on menetelmä, jossa kolmiulotteinen kappale esitetään kaksiulotteisena niin, että siitä näkyy kolme sivua.
- Perustuuko työkalu WebGL:ään, HTML5-alustaan, DOM:iin vai onko mahdollista vaihdella näiden välillä. Tässä tapauksessa keskitytään vain HTML5 – yhteensopivuuteen.
    - WebGL on yhteensopiviin selaimiin sisällytetty JavaScript-pohjainen ohjelmointirajapinta, joka renderöi interaktiivisia 3D- ja 2D-grafiikoita.
    - HTML5 on internetissä käytettävä merkintäkieli, jolla rakennetaan ja esitetään sen sisältöä.
    - DOM (Document Object Model) on kielestä riippuvainen menetelmä objektien näyttämiseen ja vuorovaikuttamiseen HTML-tyyppisissä dokumenteissa.
- Dokumentaation ja oppaiden laatu.
- Työkalun kehityksen aktiivisuus.
- Yhteisön koko (laskien esim. keskustelufoorumien viesteistä)
- Onko työkalu matalan tason vai korkean tason työkalu?
    - Matalan tason työkalu vaatii paljon koodia mutta joustaa hyvin.
    - Korkean tason työkalu vaatii vain vähän tai ei ollenkaan koodia, mutta ei ole niin joustava kuin matalan tason työkalu.
- Renderöintinopeus eli kuinka hyvin peli pyörii esimerkiksi (hitaassa) älypuhelimessa?
- Työkalun hinta (jos kaupallinen versio on saatavilla).

Taustateoriana syvennytään Suomen pelinkehityksen historiaan.

2.2 Tutkimus-/kehittämismenetelmät

Tutkimusmenetelmänä toimi työkalujen henkilökohtainen testaus ja tutkiminen sekä tutkimusten dokumentointi. Kaikki havainnot kirjattiin ylös, ja lisäksi käytettiin saatavuuden mukaan jokaisen työkalun keskustelufoorumia tai oltiin parhaimmassa tapauksessa pelimoottorin kehittäjään/kehittäjiin yhteydessä ja tutkittiin, miten työkaluista voi saada mahdollisesti enemmän irti kuin mitä alkuasetelmasta tuli ilmi. Kyseessä on siis enimmälti kvalitatiivinen tutkimusmenetelmä, joka mahdollisti erilaisia ja monityyppisiä ratkaisuja, vaikka valinnan vapaus saattoi muodostua taakaksi. Kuitenkin kaikki tutkiminen tehtiin perustuen siihen, mitä työkalu antoi käyttää ja mitä saatiin tietyn ajan sisällä tutkittua, ettei määrällisesti tutkimuskohteiden määrä olisi jäänyt liian vähäiseksi.

Aineistoa löytyi internetistä sekä sen kautta viitatuista lähdeteoksista, ja työkalujen omat oppaat olivat toimineet käyttöönoton perustana, jolloin eräänlainen opetuspuoli saatiin irti jokaisen pelimoottorin käytöstä. Lopullinen työ vain esittelee pelimoottoreita ja niiden käyttöä visuaalisen käyttöliittymän kautta, kuitenkin painottaen Javascript-ohjelmointikieltä, sillä Javascript on tarpeellinen muissa Tietojenkäsittelyn koulutusohjelmassa käytävissä opintojaksoissa. Se on myös edelleen yleisimmin käytetty ohjelmointikieli modernissa pelinkehityskulttuurissa, vaikka moni muukin kieli nostaa päätään.

2.3 Tutkimuskysymykset

Tässä esitellään tutkimuskysykset ja ja niitä tarkentavat lisäkysymykset, joihin vastataan enimmälti Työkalujen vertailu -osiossa. Näitä pidettiin mielessä työkaluja tutkiessa, ja osittain ne muodostuivat tarvekartoituksen myötä.

1) Miten kyseinen työkalu toimisi TIKON opetuskäytössä?
    a) Kuinka työkalu tarjoaa ominaisuuksiaan?
    b) Kuinka paljon työkalussa on tukea HTML5-pohjaiselle soveltamiselle?
    c) Täytyykö ko. työkalulla siirtyä alustakohtaiseen tutkimukseen jossain vaiheessa?
    d) Rajautuuko työkalun toiminta vain tietyn tyyppisiin peleihin, vai voidaanko sitä hyödyntää useammassa pelityypissä
2) Löytyykö työkalusta jotain, mitä muut tutkittavat pelimoottorit eivät tarjoa?
    a) Voidaanko tätä käyttää opetuksessa hyödyksi?
    b) Onko kyseinen toiminto rajoittava tekijä?
3) Kuinka laaja jakelukanava on mahdollista työkalulla?
4) Mitä pelimoottoria opinnäytetyön laatija opiskelijana tahtoisi nähdä opetuskäytössä, eli mikä pelimoottori tukee parhaiten IT-osaamisen oppimista?

3 Tutkimuksen toteutus

Työkaluja valittiin neljä kappaletta, kriteerinä toimien HTML5-tuki ja Open Source eli avoin lähdekoodi tai vaihtoehtoisesti ilmainen mutta mahdollisesti karsittu versio. Kaikki nämä työkalut käytiin läpi kaikkia mahdollisia toimintoja hyödyntäen, ja kun jokainen moottori oli testattu, vertailtiin niiden toimintaa toisiinsa ja tutkimustuloksena syntyi työkalujen esittely- ja vertailutyö. Tämän lisäksi

kartoitettiin tietojenkäsittelyn koulutusohjelman opettajien tarpeita aiheeseen liittyen.

Jokaisen valitun työkalun tutkimiseen ja testaamiseen käytettiin vähintään 50 tuntia, esittelyihin käytettiin 10 tuntia pelimoottorilta, ja teorian keräämiseen ja hyödyntämiseen kului n. 50 tuntia.

Tiedonkeruu tapahtui itse tutkimalla ja opettelemalla. Kaikki havainnot ja huomiot kirjattiin ylös.

Kvalitatiivisen eli laadullisen tutkimuksen käytön valintaa tuki nimenomaan työkalujen käytön vaihtelevuus, jolloin täytyi panostaa muutamien työkalujen tutkimiseen enemmän kuin määrällisessä eli kvantillisessa tutkimuksessa olisi voitu päästä. Tässä tapauksessa kvantillinen tutkimus olisi tuonut useamman työkalun tutkittavaksi, mutta lyhyemmällä käsittelyajalla, joka pidemmällä tähtäimellä ei välttämättä olisi tukenut opinnäytetyön ajatusta.

3.1 Tutkimuksesta odotettavat tulokset

Tutkimuksen lopullinen tulos on testattujen työkalujen esittely ja vertailu, jonka myötä suositellaan yhtä työkalua.

Tuloksista odotettiin selkeitä ja valmentavia, jotta Jyväskylän Ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma voisi hyödyntää tuloksia alkaessaan opettaa laajemmin pelinkehitystä HTML5-pohjalla.

Tulokset tähdättiin yleistettäviksi, jotta pelimoottorien käyttö onnistuisi kaikilla, ja että jatkotutkimus mahdollisille alustakohtaisille kehityksille olisi tehtävissä. Tämän tutkimuksen tarkoitus on luoda vakaa pohja pelimoottorien käytölle.

Opinnäytetyöstä odotettiin sekä osaamisen näyttäviä, että sitä kehittäviä piirteitä, joita voidaan hyödyntää myöhemmin mahdollisesti työllistymistarkoituksissa.

Tulokset raportoitiin kirjaamalla ensin jokainen testattu työkalu, sen ominaisuudet ja käyttötavat. Lopuksi suoritettiin raportissa vertailu, jolla työkalut eriteltiin ja varmistettiin, että JAMK:n tietojenkäsittelyn koulutusohjelmassa voidaan antaa selkeä kuva siitä, mitä työkalua tai – kaluja voidaan hyödyntää opetustarkoituksessa.

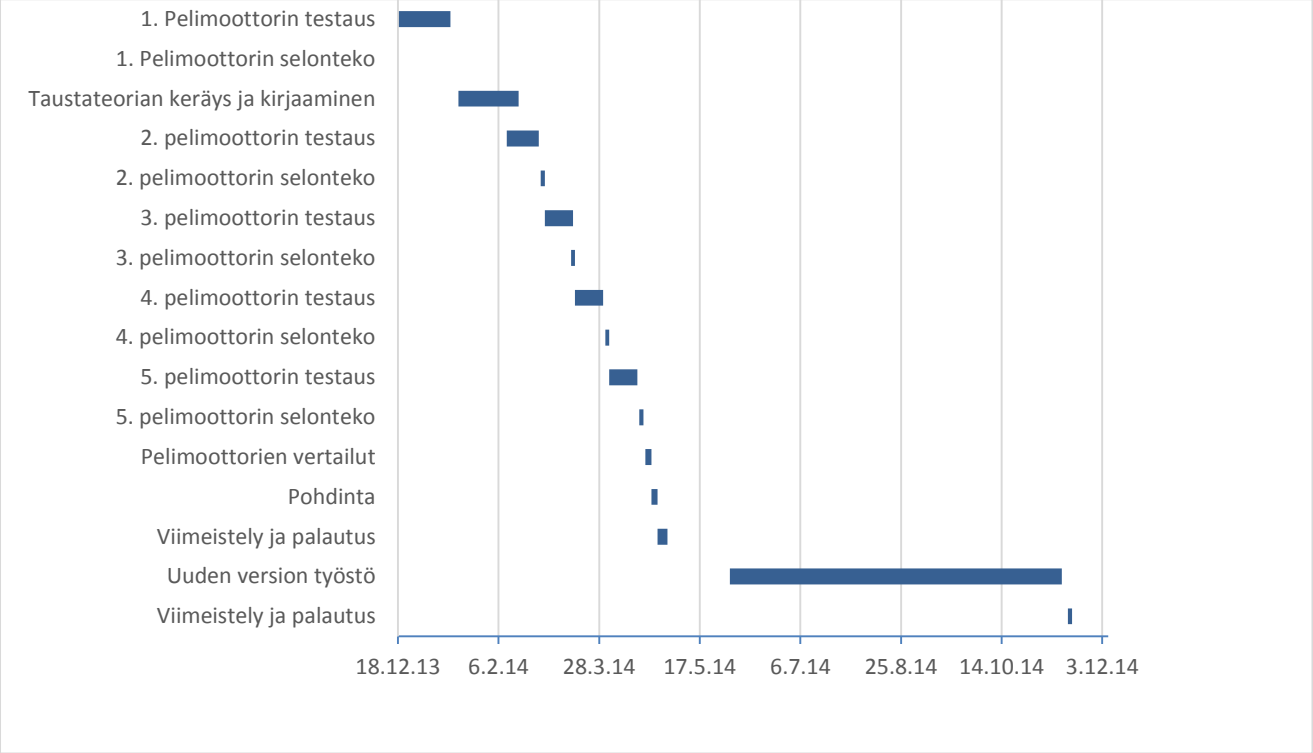
Tutkimuksessa tultiin hyödyntämään myös jokaisesta työkalun saatavilla olevaa kirjallisuutta, tarkoittaen lähinnä käyttöoppaita.

Taustateoriassa hyödynnettiin teoksia joissa käydään läpi pelin- ja sovelluskehityksen historiaa ja digitaalisen jakelun perusteita.

3.2 Aikataulu

Opinnäytetyön tekeminen aloitettiin heti tutkimussuunnitelman hyväksynnän ja aloituspalaverin jälkeen, marraskuussa 2013. Sen kuului valmistua noin neljästä viiteen kuukaudessa aloituksesta, jolloin voitiin puhua maalis-huhtikuun 2014 vaihteesta viimeisenä takarajana, jotta opinnäytetyö ehdittiin käsitellä ja mahdollisesti hyödyntää tietojenkäsittelyn koulutusohjelmassa syksyä 2014 silmällä pitäen, jolloin laajemman pelinkehitysopetuksen on määrä alkaa. Aikataulussa tapahtuneet muutokset kuitenkin viittasivat huhtikuun puolelle, kuitenkin ennen toukokuun palautuspäivää 12.5.2014. Tämä tavoite saavutettiin, mutta työtä ei hyväksytty ensimmäisellä kerralla.

Uutta versiota tehdään kesän sekä syksyn 2014 aikana, ja sen on tarkoitus olla saatavilla joulukuussa 2014 tarkistuksessa ja mahdollisesti hyödynnetään käytössä tietojenkäsittelyn koulutusohjelmassa myöhemmin.



KUVIO 1. GANT-kaavio aikataulun jäsentämiseksi

4 Suomen peliteollisuuden historiaa

Suomen nykyinen menestys pelitoimialalla on kehittynyt monen tekijän kautta jo 1980-luvulta lähtien. Näitä tekijöitä ovat mm. harrastuneisuus ja demoskenen syntyminen sekä Nokian ja puhelinoperaattoreiden osuus lähempänä vuosituhannen vaihdetta.

Kaksituhattaluvulla suomalainen pelikehitys on saavuttanut merkittävän roolin myös kansainvälisessä pelibisneksessä. Max Payne- ja Angry Birds – tyyppiset

menestyskonseptit ja Supercellin osakkeista tehdyt miljoonakaupat ovat vain näkyvin osa sitä nopeasti kasvavaa pelikehityskulttuuria, jonka piirissä Suomessa työskentelee niin harrastaja

4.1 Johdatus Suomen pelitoimialaan sekä Tekesin selvitykseen

Suomen pelitoimiala on menestynyt kansainvälisesti, ja ala on myös hyvä esimerkki elinkeinoelämän uudistumisesta ja sen rakenteen muutoksesta. Jatkuva työskentely sekä panostaminen digitaalisen teknologian osaamiseen ovat yhdistäneet erinäisten alojen asiantuntijuutta monipuolisesti. Erikoistunut osaaminen on uusien käyttäjälähtöisten innovaatioprosessien ansiota. Liiketoimintamallit, jotka käyttävät digitaalisuutta hyödyksi ja jotka yhdistetään huipputason osaamiseen, sijoittavat suomalaiset yritykset maailmanlaajuisten markkinoiden arvoverkostoihin varsin menestyksekkäästi. Pelitoimialalla tuottaa ympärilleen myös muuta liiketoimintaa, kuten oheistuoteteollisuutta ja teemapuistoja. Myös elokuva- ja animaatiotuotannossa sekä muissa interaktiivisissa median tuotteissa sekä innovatiivisissa palveluissa hyödytään pelitoimialasta. Vuosituhannen alusta alkaen peliteollisuutta on pidetty viihdeteollisuuden nopeimmin kasvaneena alueena. Vuonna 2013 sen arvo lähentyi maailmanlaajuisesti 100:aa miljardia euroa. (Isbom, Nordgren, Korhonen & Heikkinen 2013, 6.)

Suomen menestys mm. Angry Birds- ja Alan Wake -pelien myötä todistaa peliyritysten osaamisen ja luovuuden olevan meillä maailman huippua. Kokonaisliikevaihto Suomen peliteollisuudessa vuonna 2012 ylitti 300 miljoonan rajapyykin, ja miljardin rajaa lähestytään, ellei sitä olla jo ylitetty. Liiketoimintamallit ja teknologiat pelialalla ovat kokeneet rajuja muutoksia viime vuosien aikana. Digitaalisen jakelun, mobiilialustojen kehityksen, sosiaalisen median, sukupuoli- ja

ikäjakauman, ansainta- ja jakelulogiikan muutokset sekä pelillisten sovellusten ja peliteknologioiden lisääntyminen ohjelmistopohjaiseen liiketoimintaan ovat tuoneet uusia haasteita ja liiketoimintamahdollisuuksia. (Isbom ym. 2013, 6.)

Rovion, Remedyn, Fingersoftin, RedLynxin ja Supercellin tapaisia menestystarinoita toivotaan lisää, jotta kansainvälinen kiinnostus suomalaista peliosaamista kohtaan voi yhä kasvaa. Tämä näkyy muun muassa siinä, että globaalit pelijätit ovat perustaneet ja perustavat yhä uusia yksiköitä Suomeen. Vaikka peliteollisuus on vientiteollisuutta ja myös nopeimmin kasvava luovan teollisuuden ala Suomessa, sen kasvu on silti myös suuri haaste.

Tekes – teknologian ja innovaatioiden kehittämiskeskus, joka rahoittaa kehitys- ja tutkimusprojekteja, on jo vuosia rahoittanut lukuisia peliteollisuutta edistäviä hankkeita. Vuonna 2013 Neogames Finland ry. Yhdistys laati selvityksen Suomen pelialan kehityksestä 1980-luvusta aina nykypäivään saakka. Neogamesin tavoite on edistää pelitoimialan kasvua ja kehitystä, ja se on toiminut koko toimintansa ajan vuorovaikutuksessa suomalaisen pelialan kanssa. (Hiltunen, Latva & Kaleva 2013, 7.)

4.2 Ensimmäisten kotitietokoneiden vaikutus 1980-luvulla

1980-luvun alkupuolella kotitietokoneiden tulo kuluttajille mahdollisti nykyistä muistuttavan pelitoimialan synnyn. Elektronisia pelejä oli jo tehty aikaisemmin, mutta kehitys oli kallista ja pelit riippuvaisia laitteistoista. Pian ensimmäisten kotikoneiden ilmestyttyä alkoi pelien harrastepohjainen kehitys. Ensimmäiset nimenomaan kotikäyttöön tarkoitetut tietokoneet, kuten Atarin eri versiot, Commodore VIC-20 ja Commodore 64, Sinclairin mallit ja Spectravideo, olivat vaatimattomia tehoiltaan. Silti käyttäjien oli mahdollista aloittaa oma ohjelmistokehitys. Alkuvaiheessa kehittäminen harrastuksena oli järjestäytymätöntä

ja pelien kehitys tapahtui pienissä itsenäisissä ryhmissä tai yksittäisten kehittäjien toimesta. (Hiltunen ym. 2013, 10.)

Jo hyvin varhaisessa vaiheessa kuitenkin todistettiin vaatimatonta kaupallista markkinaa, joka oli kuitenkin pientä eikä tämän uuden toimialan kypsyys riittänyt vielä laajemman yritystoiminnan synnyttämiseen. Kuitenkin jotkut suomalaiset yritykset, kuten Amersoft, kokeilivat pelinkehitys- ja julkaisutoimintaa. Ensimmäiset kaupalliset pelit julkaistiin vuosina 1984−1985. Vuonna 1986 julkaistiin ensimmäinen kansainvälisille markkinoille tarkoitettu suomalainen kaupallinen peli, Starvros Fasoulasin Sanxion. (Hiltunen ym. 2013, 10.)

Pelinkehityksen laajuus ja taloudellinen merkitys jäivät 1980-luvulla pieneksi. Tämä ei kuitenkaan estänyt suomalaisen demoskenen ja pelinkehityskulttuurin pohjan syntymistä. Erilaiset kokeilut antoivat myös kokemusta pelinteosta ja pelimarkkinoiden lainalaisuuksista. Vahvaa harrastuneisuutta pelikehityksessä ja sen merkitystä pelialan kehitykselle on usein aliarvioitu. Teknologiaorientoituneeksi alaksi pelitoimiala on myös hyvin luova ala. (Hiltunen ym. 2013, 10.)

4.3 Demoskenen synty 1990-luvulla

Siirryttäessä 1990-luvulle harrastustoiminta alkoi saada uusia olomuotoja. Teknologian kehitys mahdollisti teknisesti ja sisällöllisesti vaativampien pelien kehittämisen, mikä loi harrastajakehittäjille tarpeen erikoistua tiettyyn pelinkehityksen osa-alueeseen sekä verkostoitumiseen muiden alan harrastajien kanssa. Kokoontumistarpeisiin vaikuttavia tekijöitä olivat myös tiedostojen siirron ja sähköisen ylläpidon hinta, hitaus ja hankaluus. Tämä johti usein fyysisten kopioiden jakeluun. (Hiltunen ym. 2013, 10.)

Kun harrastajien keskeistä ryhmäytymistä alkoi esiintyä yhä enemmän, lisääntyi myös näiden ryhmien välinen kilpailu. Nykyään voitaisiin pitää erikoisena, että pelien kopiosuojauksia kiertävillä alan harrastajilla eli ns. kräkkereillä oli oma osuutensa demoskenen syntymiseen. Tämä ilmeni muun muassa omien tunnuksien lisäämisellä kaupallisten pelien alkuun. Tämä johti kilpailuun lisäilyjen paremmuudesta, ja lopulta nämä lisäykset erotettiin omiksi ohjelmikseen, niin kutsutuiksi introiksi. Kilpailun kannalta introjen esittely oli helpointa tehdä kokoonnuttaessa. Tämä johti suoraan ensimmäisiin demopartyihin ja demokilpailuihin. Ensimmäinen Suomen demokilpailu Byterapers Grendelparty järjestettiin tiettävästi vuonna 1988. (http://www.byterapers.com/gallery/b1988/index.htm)

Kuitenkin koko Suomen pelitoimialan kehityksen ja demoskenen merkittävin tapahtuma on ollut ja on yhä vuodesta 1992 asti vuosittain järjestetyt Assemblyt.

1990-luku demoskeneineen toi pelinkehitykseen harrastustoimintana yritystoimintaan liittyviä elementtejä, kuten tiimit, tavoitehakuisuus, kilpailu ja säännönmukaisuus. Harrastajien järjestäytyminen demoskenen myötä on ollut ja on yhä ainakin osittainen edellytys peliyritysten syntymiselle. (Hiltunen ym. 2013, 13.)

4.4 Pelitoimialan liiketoiminnan käynnistyminen 2000-luvulla

Vuosituhannen vaihteessa pelialan liiketoiminta käynnistyi. Peliyrityksiä oli kymmenkunta ja ala työllisti vähän alle 200 ihmistä. Tämän ajanjakson aikana perustettuja yrityksiä, jotka yhä olemassa ovat muun muassa RedLynx (2000) sekä Remedy (1995). (Hiltunen, ym. 2013, 15)

Koska vuosituhannen vaihteen tilanteesta ei ole tehty varsinaista yritysselvitystä. ei pelitoimialan työllistävyydestä, liikevaihdosta tai edes yritysten lukumäärästä ei ole saatavilla tarkempaa tietoa. Vuosituhannen vaihde oli kuitenkin nopean teknologisen kehityksen aikaa. Internet teknologioineen ja talouksineen kehittyi ripeästi. Vuosia 1995–2000 on kutsuttu ICT-sektorilla taloudellisten spekulaatioiden vuoksi ”Internet-kuplaksi” tai ”dot-com-hypeksi”. Internet-kupla ei kuitenkaan ollut suorassa syy-yhteydessä pelien kanssa, pelit kun eivät sitä luoneet. (Hiltunen, ym. 2013, 15)

Suomalaisessa viitekehyksessä ilmiö oli kuitenkin mittava. Täällä se ilmeni mobiilipainoitteisuudessa: WAP-protokollan kehitys, johon Nokia otti voimakkaasti osaa, antoi suunnan suomalaisen pelialan painotuksia tulevaan. Vaikka WAP:ia mainostettiin mobiilin internetin käytön helpottavana tekijänä, se ei muun muassa heikon käytettävyyden ja tiedonsiirron korkeiden hintojen takia kasvanut taloudellisesti kannattavaksi ilmiöksi. Nokian, operaattoreiden ja rahoittajien panostuksen myötä mobiilin pelaamisen tulevaisuus vaikutti vahvalta jo 2000-luvun alkupuolella. Ensimmäiset Free-to-Play-kokeilut, pelien muuttuminen palveluiksi ja IP:hen, Intellectual Property eli immateriaalioikeuteen perustuvien brändien luonti ovat pohja, josta nykypäivän pelaaminen vuodesta 2010 eteenpäin on kotoisin. (Hiltunen, ym. 2013, 15)

Kun 2000-luvun alun kupla puhkesi, moni suomalainen pelinkehittäjä loi tuotteita, joita myytiin premium-mallilla. Tarkoituksena oli yleensä luoda immateriaalioikeus tai rakentaa brändiä. Kuitenkin immateriaalioikeuden rakentaminen tapahtui usein lyhytnäköisesti, keskittyen vain myynnin luomiseen, vaikka potentiaalisesti pidemmän aikavälin mahdollisuutena olisi ollut vahvaan immateriaalioikeuden hallintaan jatko-osien muodossa. (Hiltunen, ym. 2013, 16)

Yhtenä hyvänä esimerkkinä mobiilin peliyritystoiminnan huonosta puolesta voidaan pitää tapausta Riot-E eli Riot Entertainment Ltd. Tämä vuonna 2000 perustettu suomalainen mobiiliviihteen yritys omasi 20 miljoonan euron riskirahoituksen. Se työllisti jopa 100 henkeä ja omisti toimipisteitä Singaporessa, Los Angelesissa, Lontoossa, Roomassa, Pariisissa ja Berliinissä. Yhtiö neuvotteli itselleen merkittäviä sisältölisenssejä, kuten Taru sormusten herrasta ja Hämähäkkimies. Lisenssipelit eivät kuitenkaan tuottaneet riittävästi, ja holtiton taloudenpito johti Riot E:n konkurssiin 19. marraskuuta 2002. (Hiltunen, ym.2013, 16)

Koko pelialan painopiste ei kuitenkaan jäänyt mobiilituotantoon painottuvaksi, vaikka yrityksissä kokeiltiin mobiilituotteita. Yritystoiminnan vaihtelevuus ilmeni jo varhain yhtenä Suomen pelialan merkittävänä piirteenä. Yhtäällä nähtiin mobiili-tuotannon pelialustat, toisaalla online-tuotannon pelialustat. (Hiltunen, ym. 2013, 16)

Vuosituhannen vaihteessa tapahtui merkittäviä strategisia tapahtumia. Mobiilikokeilut suuntasivat koko pelitoimialaa kohti mobiilia pelinkehitystä. Tämä voimistui yhä enemmän muun muassa Nokian roolin vahvistumisen myötä. Toisaalta myös investointien epäonnistuminen sai sijoittajat varovaisiksi, ja seuraavina vuosina investointien saanti oli vaikeampaa. (Hiltunen ym. 2013, 16)

Internet-kuplan ja mobiilikuplan puhkeamisen myötä sekä ICT-sektorille että pelialalle saatavat investoinnit romahtivat lukumäärässä. Pelitoimiala on todettu aina hankalaksi sijoituskohteeksi. Epäonnistuneet investoinnit, korkea riski uusissa investoinneissa sekä yleinen varovaisuus sijoittajien taholla käytännössä lopetti vuosituhannen alussa investoinnit. Tämä vaikutti selkeästi pelialan rakenteisiin. Varovainen orgaaninen kasvu ja projektiliiketoiminta korvasivat yritysten maailmanvalloituksen. Mobiilipuolella tehtiin yhteistyötä mobiilioperaattorien ja

peliportfolioita operaattoreille kokoavien välittäjien eli aggregaattoreiden kesken. (Hiltunen, ym. 2013, 17)

Vuosina 2002–2006 voimakkaimmin kasvava suomalainen yritys oli Habbo Hotellia kehittänyt Sulake. Tämä online-tuotannon puolella toimiva yritys säilytti vahvan rahoitustaustansa hyvin, ja vuosina 2002–2006 sekä sen jälkeenkin Sulake oli Suomen merkittävin pelialan yritys, mitä liikevaihtoon ja työllistävyyteen tulee. Vaikka investointeja ei käytännössä ollut saatavilla vuosina 2002–2004, Nokian N-Gage sekä N-Gage QD (vuosina 2003–2004) loivat mahdollisuuden mobiilipelien kehityksen jatkumisen Suomessa. Kuitenkaan kumpikaan näistä Nokian laitteista ei ollut myyntimenestys. (Hiltunen, ym.2013, 18)

Kuitenkin Nokian investoinnit N-Gage – laitteisiin ja palvelun pelitarjontaan sekä samaan aikaan kehittynyt Java-pohjaisten pelien maailmanmarkkinan kasvu vahvistivat omalta osaltaan Suomen pelialan mobiilisuuntautuneisuutta. Ne olivat valmistelevia tekijöitä koko toimialan myöhempää digitaalisen jakelun markkinaa varten. Investointilama suuntasi omalla tavallaan Suomen pelinkehitystä kohti pieniä yrityskokoja ja laitealustoja, joiden pelinkehityksen kynnys oli matala ja joihin keskittyvät projektit olivat maltillisia kustannuksiltaan. Tällöin pieniä investointeja vaativa pelinkehitys oli mahdollista. Toisin kuin Ruotsissa, Suomessa pelit keskittyivät isojen konsolipelien sijasta mobiilipeleihin. (Hiltunen, ym. 2013, 18)

Nokian vahva rooli muun muassa N-Gagen kautta avasi pelinkehittäjille uuden mobiilin liiketoiminnan mahdollisuudet. Suomalaisille pelinkehittäjille oli selkeää etua yhteistyökumppanin kotimaisuudesta. Suomen mobiilipeliteollisuuden kasvu herätti ulkomaalaisten toimijoiden kiinnostuksen ja tämä johti myös yrityskauppoihin. (Hiltunen, ym. 2013, 18)

5 Pelinkehitys ja -jakelu nykypäivänä

Nykyinen peliteollisuus on saanut koti- ja käsikonsolien ja niiden pelien lisäksi rinnalleen mobiilipelit ja digitaaliset jakelukanavat. Tämän vuoksi myös pelinkehitys on muuttunut huomattavasti, oli sitten kyse kehitysmallista tai jakelukanavista.

5.1 Pelinkehitysmallit

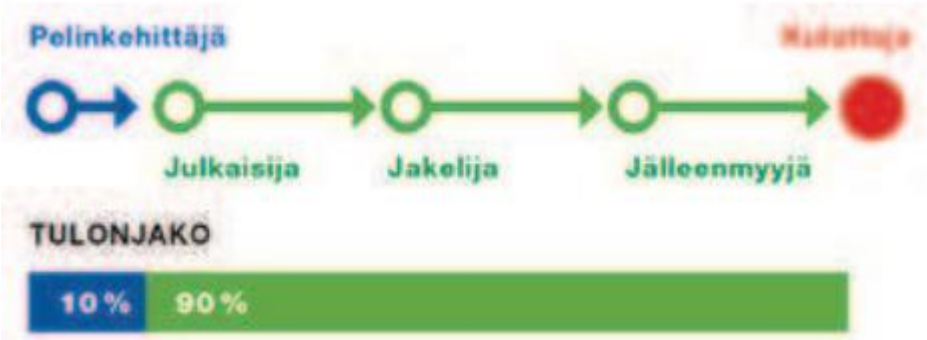
Digitaalisen jakelun luomat mahdollisuudet ovat laajentaneet Suomen pelitoimialaa yhä entisestään. Ennen digitaalisen jakelun aikakautta pelialalla on ollut voimassa vahvasti fyysiseen jakeluun perustuva vähittäiskauppa.

Perinteinen pelinkehitysmalli koostuu seuraavista vaiheista (Suomen Pelialan Strategia 2010–2015. Visio 2020. Suomen Pelinkehittäjät Ry):

1) Pelinkehittäjä dokumentoi konseptin ideastaan
2) Dokumentin pohjalta tehdään pelin demo
3) Demon avulla tuotantoa ja IP-oikeuksia markkinoidaan julkaisijalle
4) Julkaisija maksaa pelinkehityksen hyväksyttävillä toimituksilla
5) Julkaisija julkaisee pelin, hoitaen sen markkinoinnin ja peli jatkaa jakelijan kautta vähittäiskauppaan ja siitä kuluttajalle.

Fyysisen jakelun kustannukset sekä markkinoinnin ja jakelun vaatima rahallinen ja työmäärällinen panostus aiheuttaa sen, että pelinkehittäjillä ei ole yksinkertaisesti mahdollisuuksia hallita kaikkia arvoketjun osa-alueita, jolloin tulonjako keskittyy enemmän julkaisijoille, jakelijoille ja jälleenmyyjälle. Pelinkehittäjälle jää vain pieni osa pelinkehityksen vaihdosta. Julkaisijan taloudellisen riskin merkittävyyden vuoksi tämä yleensä vaatii sopimusta tehtäessä täydet oikeudet pelinteon yhteydessä syntyvän tuotteen IP-oikeuksiin ja niistä syntyviin jatkossa tapahtuviin

hyödyntämismahdollisuuksiin. (Hiltunen, Latva, & Kaleva, 2013, 19.)



Kuva 1. Perinteisen pelijakelun arvoketju. (Suomen Pelialan Strategia 2010–2015. Visio 2020. Suomen Pelinkehittäjät Ry)

Monissa tapauksissa pelinkehittäjät saavat vain pienen osuuden myynnistä tulevista tuloista. Prosenttiosuus on hyvin vähäinen, ja tulouttaminen aloitetaan vasta, kun myynti on kattanut julkaisijan kustannukset. Pahimmassa tapauksessa kehittäjä ei saa myynneistä lainkaan osuutta. Perinteinen malli asettaa pelinkehittäjän alihankkijan asemaan jopa silloin, kun peli perustuu kehittäjän omaan IP:hen. Tällöin voidaan todeta, että perinteinen malli on pelinkehittäjälle epäedullinen, koska se ei mahdollista esimerkiksi luodun IP:n jatkohyödyntämistä tai täysimittaisen hyödyn saamista mahdollisesta hittituotteesta. Myös demon tekemisen vaatima panostus on riski. Nykyaikaisen konsolipelin demon kustannukset vaihtelevat 0,5–1,5 miljoonan euron välillä, ja mikäli julkaisusopimusta ei saada, riskin kantaa pelinkehittäjä. Malli on kylläkin turvallinen, koska tällöin pelinkehityksen mahdollistaa julkaisijan takaama rahoitus. Perinteisessä mallissa ilmenee myös ongelma, joka juontaa juurensa yritysten pienteen kokoon. Pienet kehittäjät eivät ole julkaisijan näkökulmasta parhaimpia kumppaneita pienen koon aiheuttamien riskien takia. Tämä vaikuttaa sopimuksen saamiseen tehden siitä vaikeaa. (Hiltunen ym. 2013, 19, 21.)

Tiedonsiirron ja Internetin kehitys on mahdollistanut tiedonsiirron kasvun digitaalisessa muodossa. Tämä on muuttanut pelitoimialan arvoketjua vaikuttamalla pelinjakeluun. Useimmat pelikonsolit ja -laitteet on mahdollista yhdistää internetiin, jonka myötä pelilaitteisiin sidotut online-kauppapaikat ovat saaneet merkittäviä mahdollisuuksia. (Hiltunen ym. 2013, 21.)

Kuva 2. Digitaalisen pelijakelun arvoketju. (Suomen Pelialan Strategia 2010–2015. Visio 2020. Suomen Pelinkehittäjät Ry)

Digitaalisen jakelun liiketoimintamalli suoraviivaistaa perinteisen jakelun mallia. Ero löytyy siitä, että jaettaessa digitaalisesti verkon kautta ei ole tarvetta fyysisen jakelun vaatimalle jakelija-vähimmäiskauppa-portaalle, jolloin kustannukset ovat myös paljon pienemmät kuin fyysisessä jakelussa, koska esim. levyjä ei tarvitse valmistaa. Perinteisen jakelumallin sisältämä ”suuruuden ekonomia” asettaa myös rajoituksia sille, minkä kokoisia pelejä kannattaa jakaa. Digitaalinen jakelu poistaa tämän esteen kokonaan ja näin avaa uuden markkinapaikan antaen myös pienille pelifirmoille mahdollisuuden kehittää halvalla sisältöä, ja säästöt antavat kaikenkokoisille pelinkehittäjille mahdollisuuden säilyttää IP-oikeudet itse kehittämäänsä tuotteeseen, mikä taas johtaa IP:n täydelliseen jatkohyödyntämiseen. (Hiltunen ym. 2013, 21.)

Digitaalinen jakelu on muuttanut myös mobiilipuolella. Muutos kuitenkin tapahtui eri tavalla. Vuonna 2007 Java-ohjelmointikieli oli merkittävin kehitystyökalu mobiilipelien kehityksessä. Javan suurin ongelma oli sen skaalautumattomuus ja päätelaitekannan pirstaloituneisuus. Jokaisesta pelistä oli pakko tehdä oma versionsa jokaiselle päätelaitteelle, jolle se aiottiin tuoda. Käytännössä tämä tarkoitti jopa tuhansia eri versioita, jotta peliä olisi saatu jaettua mahdollisimman suurelle pelaajakunnalle. Kustannusten vuoksi tällainen pelinkehitys varsinkin suomalaisessa mobiilipeliyrityksessä oli mahdotonta voiton keräämisen kannalta. (Hiltunen ym. 2013, 21.)

Vuonna 2007 myös teleoperaattoreilla oli valtava osuus mobiilipelien jakelussa. Pelit ladattiin operaattorien palvelun kautta. Operaattori veti välistä huomattavan osuuden pelin hinnasta voittona, ja jakelukanava oli kömpelö. Nykyinen jakelu, joka tapahtuu internetin kautta, on kuluttajan kannalta paljon miellyttävämpi käyttää. Pelinkehittäjille on helpottavaa, että nykyään pelit tehdään tietylle päätelaitteelle ja jaetaan päätelaitekohtaisten kanavien, kuten Applen Appstoren ja Googlen Android Marketin, kautta. (Hiltunen ym. 2013, 21.)

Pelinkehittäjän kannalta toinen tervetullut uudistus on suoraan jakelukanavaan integroitu maksumekanismi. Se on muuttanut mallia, jolla voittojen jako tapahtuu, kun verrataan esim. aiempaan operaattorien säätelemään mobiilipelimarkkinaan. Online-pelit eivät ole kuitenkaan kokeneet samoja muutoksia. Esimerkiksi Sulakkeen Habbo Hotel, joka oli ennen Roviota ja Supercelliä suurin suomalainen pelituote liikevaihdolla mitattaessa, on toiminut alusta alkaen online-mallissa. Tällä sektorilla digitaalinen jakelu ei siis aiheuta minkäänlaisia, ainakaan merkittäviä, muutoksia. (Hiltunen ym. 2013, 22.)

Kaiken kaikkiaan digitaalinen jakelu on mahdollistanut PC-, konsoli- ja mobiilipeliteollisuudelle monta asiaa. Omaan IP:hen pohjautuvan pelin

kehittäminen ja IP:n oikeuksien säilyttäminen sen luoneessa yrityksessä, pienemmät tuotantokustannukset, jotka mahdollistavat pienempienkin pelinkehittäjien pääsyn markkinoille, myyntimenestyksen täysimittainen hyödyntäminen, suurempi vapaus jakelukanavan ja yhteistyökumppaneiden valinnassa, erilaisten liiketoimintamallien hyödyntäminen, uudet ansaintamallit sekä pelien tuottaminen hyvällä panostus/tuotos-suhteella ovat merkittävimmät esimerkit. (Hiltunen ym. 2013, 22.)

Vaikka digitaalinen jakelu on avannut pelinkehittäjille uusia mahdollisuuksia, myös riskejä löytyy. Ilman julkaisijaa toimivat yritys joutuu vastaamaan tuotannosta sekä markkinoinnista ja niiden tuomista kustannuksista. Mahdollisuuksien myötä riskitkin ovat kasvaneet ja kehittäjien toimintakenttä on laajentunut. Juuri näiden riskien takia jotkut kehittäjät pysyttelevät edelleen perinteisessä julkaisijapohjaisessa mallissa. Kuitenkin mallien käyttö on strateginen asia, ja ”yhtä oikeaa” toimintamallia ei ole. (Hiltunen ym. 2013, 22.)

5.2 Digitaaliset jakelukanavat

Ensimmäiset digitaaliset jakelukanavat katsotaan syntyneen jo 2000-luvun vaihteessa mm. mobiilipelien myötä. Niiden kautta jakelu siirtyi PC-alustalle, josta se on lopulta levinnyt kattamaan kaikki pelialustat, myös konsolit.

Jakelukanavia ovat PC- ja Mac-päätteillä Valven vuonna 2003 julkaisema Steam, Electronic Artsin vuonna 2005 julkaisema ja vuosina 2006, 2007 sekä 2011 korvattu EA Origin, aikaisemmilla nimillä EA Downloader (2005), EA Links (2006) sekä EA Store ja EA Download Manager (2007). (Leigh Alexander, 2007.) Lisäksi Ubisoft aloitti vuonna 2009 oman Uplay-palvelunsa omille peleilleen. (Good 2009).

Jokaisella suurella konsolikehittäjällä on omat jakelukanavansa. Microsoftilla on Xbox-konsoleissa vuonna 2004 julkaistu Xbox Live Arcade, Sonylla PlayStation-

konsoleissa vuodesta 2006 asti toiminut PlayStation Network ja Nintendon Wii-konsoleilla vuodesta 2008 käytetty WiiWare, joka on myöhemmin muuttunut eShopiksi WiiU- ja 3DS-konsoleille. (Hiltunen, ym. 2013, 23.)

5.3 Mobiilijakelukanavat

Mobiilijakelukanavina voidaan mainita Applen Appstore (2008), Android Market (2008), Nokia Store (2007) sekä Windows Marketplace (2009).

Mobiilijakelukanavilla on omia erityispiirteitään, joilla se houkuttelee kehittäjiä ja julkaisijoita. Applen Appstore yhdistää laajat käyttäjämassat, hyvät monetisaatiomahdollisuudet ja tehokkaat päätelaitteet. Verrattaessa Android-laitteiden laajempaan käyttäjäkuntaan ja suurempaan latausmäärään Applen valttikortti on sen laitteiden käyttäjien rahankulutus maksullisiin sisältöihin. Applen osuus kaikista maksullisista applikaatiolatauksista on 74 % ja Androidin 20 %. (McCraken 2013.)

Googlen mainosyhtiön luonteen vuoksi Googlen Android-alustan sovelluksissa tähdätään suureen latausmäärään mainosten näkyvyyden varmistamiseksi. Tämä ikävä kyllä johtaa laadunvalvonnan heikentymiseen, jolloin jakelukanavaan pääsevät sisällöltään huonot tai jopa toimimattomat sovellukset, jotka poistetaan vasta jälkikäteen. Myös laiteyhteensopivuus on ongelma. (Hiltunen ym. 2013, 25.)

Microsoft Windows Phone -alustan laitteiden käyttäjäkunta on heikko verrattuna iOS- ja Android-alustan laitteisiin, alle 5 % (Lunden 2012). Tällöin myös jakelukanavan latausmäärä on alhainen. Sovellustarjonta on vähäisempää kuin muilla kanavilla, ja niiden vaihteleva laatu hidastaa laitemyyntiä. Pieni käyttäjäkunta heikentää alustan houkuttelevuutta kehittäjien silmissä. Tällöin voidaan puhua muna-kana-ongelmasta. Microsoft omalta osaltaan kuitenkin rahoittaa Appcampus-

ohjelmansa avulla Windows Phone -alustan sovelluksia, jotta ongelma saataisiin hoidettua. (Hiltunen ym. 2013, 25.)

6 Mobiilipelaamisen nousu

Tampereen yliopiston professori Frans Mäyrä on ollut mukana jo neljä kertaa Pelaajabarometri-kyselytutkimuksen toteutuksessa. Vuonna 2013 yhdessä Laura Ermin kanssa toteutettu kyselytutkimus paljastaa, kuinka mobiilipelaaminen on yhä suuremmassa roolissa varsinkin Suomen peliteollisuudessa. Tutkimukseen saatiin 972 satunnaisotannasta saatua 10–75 vuotiasta Manner-Suomen asukasta. Prosenttiluvuilla viitataan näihin vastaajiin, kokonaisvirhemarginaalin ollessa 3 %.

6.1 Digitaalisten viihdepelien kyselytutkimuksen tulokset

Digitaalisia pelejä on pelannut ainakin joskus 73,6 % vastaajista, joista aktiivisia, eli vähintään kerran kuussa pelaavia pelaajia, oli 52,5 %. Lisäksi aktiivisia naispelaajia löytyi 45,9 %, ja miesten vastaava määrä oli 59,4 %. (Mäyrä & Ermi. 2013, 15)

Vaikka pääosin suomalaiset yhä pelaavat enimmälti ei-digitaalisia pelejä, kuten tutkimukseen mukaan laskettuja Veikkauksen rahapelejä (46.5 %) ja pulmapelejä kuten sanaristikoita ja sudokuja (41,6 %), on silti mobiililaitteilla pelattavien pelien osuus merkittävässä nousussa (28.6 %). Yksin pelattavat tietokonepelit ovat myös hyvin suosittuja (27,8 %) ja vastoin ainakin omaa käsitystä, konsolipelit (18,9 %) ovat suomalaisten keskuudessa suosioltaan pihapelien tasoa (18,7 %). (Mäyrä ym. 2013, 16)

Mobiilipelien aktiivisten pelaajien, eli pelaajien jotka ovat pelanneet vähintään kerran viimeisen kuukauden aikana, osuus oli 28,6 %. Näistä viikoittaisia pelaajia oli 11,9 % ja päivittäisiin pelaajiin lukeutui 7,5 % vastaajista, jolloin noin kerran kuussa mobiilipelaajia pelaavia oli 9,2 %. (Mäyrä ym. 2013, 17)

Niiltä vastaajilta, jotka olivat viimeisen kuukauden sisään pelanneet jotain digitaalista peliä, pyydettiin kahden eniten pelaamansa pelin nimi. Vastaus tuli hieman alle puolelta kyselyyn osallistuneista (44 %). Suosituin peli on ollut viimeiset kolme vuotta pasianssipelit kuten Spider, vapaakenttä ja pasianssi (115 mainintaa). Tämän jälkeen tulevat Angry Birds (47 mainintaa), Veikkauksen selaimella pelattavat digitaaliset pelit (39 mainintaa), Mahjong-pelit (31 mainintaa) ja listan yhtenä uutena pelinä Candy Crush (25 mainintaa), jota pelataan mm. Facebookissa ja mobiililaitteilla. Kiinnostavan tästä listauksesta tekee se, että mediassa kaupallisella menestyksellä huomiota saanut Clash of Clans – peli ei sijoitu tässä listassa ainakaan 16 mainituimman pelin joukkoon. (Mäyrä ym. 2013, 21–23)

6.2 Muutoksia

Verrattaessa edellisten vuosien pelaajabarometreihin, ei digitaalisen pelaamisen kokonaisaktiivisuudessa huomattu suuria muutoksia. Vuonna 2011 aktiivisten digipelaajien osuus oli 56 % väestöstä, ja vuonna 2013 osuus oli alle 53 %, ei muutos kuitenkaan ole merkitsevä tilastollisesti. (Mäyrä ym. 2013, 29)

Suurin muutos kohdattiinkin digitaalisen mobiili- ja tietokonepelaamisen suosion muutoksiin. Älypuhelimilla ja tabletti-laitteilla pelaamisen kasvu on vahvistunut vuodesta 2011, jolloin mobiilipelaajia oli vain n 21 %: vuonna 2013 vajaat 29 % vastaajista oli aktiivisia mobiilipelaajia. Muutos on huomattava verratessa sitä ensimmäiseen pelaajabarometriin vuonna 2009, jolloin aktiivisten mobiilipelaajien määrä oli vain n. 13 %. On todennäköistä, että nämä pelaajat ovat siirtyneet yksin

pelattavista tietokonepeleistä, sillä ko. aktiivisen pelaajakunnan lasku on havaittavissa (v. 2011: 34,0 %, v. 2013: 27,8 %). (Mäyrä ym. 2013, 30)

Voidaan siis todeta, että digitaalisen jakelun noustessa vallitsevaksi tavaksi jakaa sisältöä tietokoneisiin, pelikonsoleihin ja mobiililaitteisiin ei enää tavallinen fyysisen kopion ostaminen ole tavallisin tapa hankkia pelejä. Mobiilipelaaminen on selkeästi kasvussa.

7 Teorian yhteenveto

Jotta suomalaisen pelitoimialan kasvu ja menestys olisi varmistettu myös tulevaisuudessa, on meidän vastattava haasteisiin, jotka liittyvät osaavan työvoiman ja rahoituksen saatavuuteen. Lisäksi yrityksille on tärkeää reagoida nopeasti toimialan ja markkinoiden muutoksiin. Tällä hetkellä Suomen peliteollisuus muodostuu pääosin mobiili- ja tablettipohjaisista peleistä, joita markkinat ovat väärällään ja jotka yleensä onkin toteutettu tavallisimmilla ohjelmointikielillä, näihin lukeutuva myös HTML5, Javascript ja C#. Tämän vuoksi opinnäytetyössäkin esitellään mahdollisimman erilaisia työkalua lähinnä HTML5-pohjaisessa pelinkehityksessä, jotta mahdollisen pelinkehityskoulutuksen kautta voidaan tarjota työnantajille väkeä, joilla on osaaminen viimeisimmillä mobiilipelien ohjelmointikielillä ja työkaluilla.

Lisäksi on tärkeää huomioida, miten nykyinen peliteollisuuden nousu on vaikuttanut siihen viittaavaan koulutukseen, jolloin yhä useammat yliopistot ja ammattikorkeakoulut tarjoavat pelinkehitykseen keskittyviä opintoja. Remedyn apulaishenkilöstöpäällikön Jukka-Pekka Parkkosen mukaan: ”(Pelinkehityksen) opinnoissa kannattaa perehtyä ohjelmointiin ja sen eri osa-alueisiin, mutta myös oma harrastuneisuus on tärkeää. Monella alalla olevilla onkin kokemusta

demopiireistä, joka on hyvä väylä tutustua alan ihmisiin.” Tällä siis voidaan viitata siihen, kuinka jo 1990-luvulla syntynyt demoskene on tärkeä tuntea pelinkehityksessä, jotta voidaan tarjota puitteita pelialan koulutuksen kasvulle.

Yhä useampi Suomessakin kehitettävä mobiilipeli rakentuu jonkin yleisessä käytössä olevan pelimoottorin päälle, jolloin on tärkeää osata niistä edes paria erilaista. Poikkeuksena sääntöön ovat tietenkin isompien firmojen suurilla budjeteilla rakennetut pelimoottorit. Vaikka niiden käyttötarkoitukset olisivat samat, kuten musiikin tai ”painovoiman” määrittämien, niin jokaisella pelimoottorilla on oma tapansa toteuttaa kyseiset toiminnot.

Jyväskylän Ammattikorkeakoulun Tietojenkäsittelyn koulutusohjelma (TIKO) haluaa varmistaa, että heidän opiskelijoilleen tarjotaan mahdollisuus oppia pelinkehitystä, mikäli kyseiselle kasvavalle teollisuudelle haluaa erikoistua. Kuitenkin tämän pelinkehityksen täytyy tukea myös muuta oppimista, kuten JavaScriptin tai muun mahdollisen ja laajalti käytetyn ohjelmointikielen käyttöä.

8 Tutkittavat työkalut

Tutkimuksen aikana käytiin läpi neljä pelinkehityksen työvälinettä, ja tulokset jakautuvat kahteen eri kokonaisuuteen: ensimmäisessä osiossa valotetaan työkaluja, niiden taustoja ja mahdollisia erikoisuuksia ja jälkimmäisessä osiossa vertaillaan näitä työkaluja, lopulta suositellen yhtä näistä työkaluista opetuskäyttöön. Suositusta tehtäessä tärkein kysymys on seuraava: ”Minkä pelinkehityksen työkalun oppiminen tukee parhaiten IT-osaamisen opiskelua?”

Työkaluja tutkittiin yksitellen, joten lopullisen vertailun helpottamiseksi tehtiin taulukko, johon kerättiin erinäisiä vertailtavia tietoja työkaluista myöhempää referointia varten. (Liite 1: Vertailutaulukko)

8.1 Construct 2

Construct 2 on Scirra-yhtiön luoma HTML5-pelien luomiseen soveltuva pelinteko-ohjelmisto, joka tarjoaa myös omat koodinsa ja pelimoottorin lähinnä 2D-pelien luomiseen. Sitä markkinoidaan astinlautana pelinkehityksen maailmaan. Ilmaisen version lisäksi Constructiin voidaan ostaa lisenssi, joka lisää työkalujen ja mahdollisuuksien määrää. Ilman lisenssiä pelejä voidaan tehdä mm. omille kotisivuilleen, Scirra Arcadeen, Facebookkiin ja Chrome Web Storeen. Lisenssi vaaditaan iOS-, Android-, Windows-, Mac-, ja Linux-kehitykseen. Opiskelijalisenssin voi ostaa oppilaitoksen koneille 21,99 euron kuukausihintaan, mutta vuoden pituinen lisenssi maksaa 194,99 euroa. Lisäksi opiskelijalle voidaan tarjota ”Take home” –lisenssi, jolla siis opiskelija saa koululta lisenssin kotikäyttöön. Tämä lisenssi maksaa 1,19 euroa oppilaalta kuukaudessa, myös vuoden pituisen ”take home” –lisenssin voi hankkia hintaan 12,99 euroa oppilaalta.

Construct 2 käyttää visuaalista käyttöliittymää, jonka kautta pelin luominen ja muokkaaminen on yksinkertaista. Pohjalle voidaan asettaa objekteja png-kuvatiedostomuodossa ja manipuloida niitä omina yksiköinään. Lisäksi Construct tukee Tilemap-kuvatiedostoja, jotka sisältävät valmiita ruutupohjalle aseteltuja objekteja, joita voi vapaasti lisätä ja muokata useita kappaleita.

Construct 2-työkalun käyttöliittymään on luotu valmiita pohjia eri tyyppisille peleille, kaikki kuitenkin 2-ulotteisten pelien luomiseen. Näihin tyyppeihin lukeutuu muun muassa tasoloikka, ylhäältä kuvattavat ammuskelut, reaaliaikaiset strategiat sekä

fysiikka-aivopähkinät. On kuitenkin toimivan yhteisön ansiota, että työkalulla voidaan luoda myös isometrisiä pelejä.

8.2 Turbulenz SDK 0.28.0

Turbulenz SDK 0.28.0 (SDK = Software development kit eli sovelluskehitys-paketti) on Open Source HTML5-kehitykseen suunniteltu työkalu. Sitä saa vapaasti muokata, sillä tehtyjä pelejä saa julkaista vapaasti ja periä maksua kyseisten pelien pelaamisesta. Turbulenz-moottorin voi myös yhdistää muihin erillisiin web-palveluihin ja teknologiaan. Turbulenzin käyttö perustuu JavaScriptin tuntemukseen.

Turbulenz sisältää sekä matalan tason että korkean tason työkalut. Matalan tason työkalut on rajattu tiettyjen peligenrejen kehittämistä varten, jolloin vapautta jatkaa koodia on vähän, kun taas korkean tason työkalut ovat hyvin muokattavissa itse, mikäli JavaScript on tarpeeksi tuttua.

Lisäksi Turbulenz tarjoaa oman alustan, jolla julkaista pelejä muiden pelattavaksi, testattavaksi ja arvioitavaksi.

8.3 Phaser

Phaser on Photon Stormin kehittämä Open Source -työkalupaketti. Photon Storm on vuonna 2006 perustettu indie-pelien kehitysryhmä, joka teki pelejä työskentelijöidensä vapaa-ajalla. Vasta vuonna 2012 muuttui täysipäiväiseksi pelifirmaksi, jonka tavoite on tehdä HTML5-pohjaisia pelejä sekä työpöytä- että mobiiliselaimille.

Phaser julkaistiin vuonna 2013, ja se on Photon Stormin taidonnäyte HTML5-pelimoottorina. Phaser on erittäin suosittu työkalu, yli 1000 github-tähteä sekä laaja, aktiivinen kehittäjäyhteisö.

Työkalupaketin käyttöä varten on luotu myös MightyEditor, joka on selainpohjainen editori. Tällä voi paitsi luoda yksinkertaisia pelejä, myös päästä käsiksi Phaserin lähdekoodiin.

Phaser-työkalupakettia voidaan käyttää myös muiden työkalupakettien kanssa.

8.4 LimeJS

LimeJS:n loi Digital Fruit vuonna 2011 ja on siitä lähtien tasaiseen tahtiin päivitetty, vaikka tahti on hiipunut viime kuukausina. Se soveltuu parhaiten kosketusnäytöllisten pelien luomiseen, mutta soveltuu myös selainpelien kehittämiseen. LimeJSn rakentamisessa on hyödynnetty Googlen rakentamaa JavaScript-kirjastoa, Closure Library:a, josta on kehitetty paljon Googlen tuotteita, kuten Gmail tai Google Docs. LimeJS on julkaistu Apache-lisenssillä, eli sitä voi vapaasti käyttää mihin tarkoitukseen haluaa, muokata ja julkaista alkuperäisenä tai muokattuna.

LimeJS:n päivitystahdin hiipuminen voi viestiä kahdesta eri asiasta: joko kiinnostus LimeJSn käyttöön on hiipumassa, tai koodi on kutakuinkin niin hiottu, että sitä voidaan hyödyntää pelinkehityksessä ilman ongelmia. Päivitykset ovat nykyään enimmäkseen ongelmien korjauksia kuin uusien ominaisuuksien lisäämistä.

9 Työkalujen vertailu

Tämän osion tarkoitus on vertailla aiemmin esiteltyjä työkaluja, jotta saadaan selville, mikä työkalu tukisi parhaiten sekä opiskelijan että opettajan kannalta opiskelijan kehitystä sekä pelin- että sovelluskehittäjänä. Jokainen kriteeri on lajiteltu omaksi alaluvukseen, jossa vertailu suoritetaan.

9.1 Mobiili- ja tablettituki

Yksi tärkeimmistä seikoista nykypäivän pelinkehityksessä on mobiililaitetuki, jotta voidaan kehittää lyhyitä ja viihdyttäviä pelejä esimerkiksi pikaiselle pelihetkelle bussipysäkillä. Jokainen esitelty työkalu tukee mobiili- ja tablettialustoja pelinkehityksessä.

Construct 2-työkalun editorilla voidaan helposti kääntää pelin ohjausmenetelmät kosketusnäytölle, kuten esimerkiksi ruudulla näkyvillä napeilla, joilla ohjata hahmoa. Turbulenz hyödyntää HTML5-tukeaan sekä tapaa työstää pelit erillisesti suoraan mobiililaitteille pelkkien selainpelien sijaan. Phaser on yksinomaan rakennettu mobiiliselaimille ja –sovelluksille, lisäksi Phaser sisältää oman skaalaus-koodin, joka skaalaa eli suhteuttaa pelikuvan koon toimimaan minkä kokoisella näytöllä sen halutaan toimivan. Myös LimeJS on luotu nimenomaan mobiililaitteita ajatellen, vaikka selainpelien kehittäminen on edelleen mahdollista.

Jokainen työkalu siis tukee mobiilipelien kehittämistä, joten vertailua parhaimman työkalun valitsemiseksi on tässä tapauksessa vaikea tehdä ilman muita kriteerejä.

9.2 Käyttöliittymän joustavuus

Joustava käyttöliittymä takaa, että pelin tekemisen aloittaminen on helppoa ja mukavaa.

Construct 2-työkalun käyttöliittymän joustavuus riippuu luotavasta pelistä, mutta sisältää pohjat mm. tasoloikalle, fysiikka-aivopähkinöille sekä reaaliaikaisille strategioille. Tietenkin muokattavuus rajoittuu näiden pohjien myötä, mutta avustavat hyvin alkuun pelinkehityksessä.

Turbulenz vaatii alleen tekstieditorin JavaScript-koodausta varten, selaimen kuten Google Chrome tai Mozilla Firefox, sekä paikallisen palvelimen ylläpitoon suunnitellun ohjelman. Tällainen sisältyykin Turbulenzin pakettiin. Lisäksi Turbulenzilla on oma selainpohjainen visuaalinen käyttöliittymä, joka voi vaatia aluksi opettelua. Kuitenkin opiskelijan kannalta on parempi, että käytetään tekstieditoria JavaScriptin luomiseen ja muokkaamiseen.

Phaser –työkalun avulla voidaan luoda ja muokata yksinkertaisia kaksiulotteisia pelejä selainpohjaisella MightyEditorilla, jolla voidaan myös katsoa ja muokata Phaserin lähdekoodia. Tietenkin myös tavallisella tekstieditorilla tämä on mahdollista. Työkalua suositellaan myös siksi, että sen koodeja voidaan käyttää muiden työkalujen kanssa.

LimeJS-työkalupaketin asennuksen mukana tulee Python-työkalu, jolla voidaan luoda sekä muokata LimeJS-projekteja, kuten JavaScript-koodia. Toisin sanoen LimeJSn käytössä on hyvä osata hyödyntää Python-työkalua.

Käyttöliittymän näkökulmasta Construct 2 on silmälle miellyttävin, sillä se on kokonaan oma graafinen käyttöliittymänsä, mutta oppimisen kannalta Turbulenz tai LimeJS ovat paremmat nimenomaan JavaScript-pohjaisen koodaamisen takia. Mitään

ylimääräisiä asennuksia ei Turbulenz:n tai LimeJS:n kanssa välttämättä tarvita, ja koodia voi tarkkailla ja muokata itse alusta loppuun.

9.3 Integroitu fysiikkamoottori

Integroidulla eli ”sisäänrakennetulla” fysiikkamoottorilla voidaan mallintaa esimerkiksi painovoimaan perustuvia puzzle-pelejä, Rovion Angry Birdsin tyyppisiä ”tuhoa linna” –pelien fysiikkaa sekä yleisesti fysiikkapohjaisia objekteja. Näiden renderöinti ruudulle vaatii prosessointitehoja, joka on mobiililaitteilla pieni verratessa tietokoneisiin.

Construct 2 sisältää integroidun fysiikkamoottorin, mutta työkalun kehittäjä Scirra suosittelee varsinkin oppaissaan rajoittamaan ruudulla samanaikaisesti näkyviä fysiikkaobjekteja renderöintisyistä.

Turbulenz tarjoaa integroidut fysiikkamoottorinsa sekä kaksi- että kolmiulotteisiin peleihin.

Phaser sisältää useita fysiikkamoottoreita, joista jokaisella on oma käyttötarkoitus. Arcade Physics käsittää nopean törmäystarkistuksen, Axis-Aligned Bounding Box –nimisen törmäyksen mallinnuksen, jonka mukaan ensin määritetään, mikäli kaksi objektia ovat jollain lailla kosketuksissa toisiinsa koordinaattiensa perusteella, jonka jälkeen siirrytään määrittämään, millainen kosketus on kyseessä ja miten se vaikuttaa objekteihin. Ninja Physics tuo mahdollisuuden käyttää monimutkaisempia muotoja sekä kaltevia pintoja, mikä sopii hyvin esimerkiksi pelin tasojen luomiseen. P2.JS on objekteille sovellettava fysiikkamoottori, joka sisältää muun muassa polygonituen ja pomppivat objektit.

LimeJS sisältää Box2D-kirjaston, joka on Open Source C++ -fysiikkamoottori. Se sisältää kaikki tarpeelliset kaksiulotteiseen fysiikkamallinnukseen tarvittavat työkalut.

Fysiikkamoottorin puolesta Phaser sisältää eniten vaihtoehtoja fysiikan mallinnukseen, mutta LimeJS on yksinkertaisin, sillä Box2D on laajalti käytetty. Kuitenkin Turbulenz on monipuolisin nimenomaan kolmiulotteisten pelien luonnin tukemisen ansiosta.

9.4 Verkkopelituki

Verkkopelituen tarvetta voidaan perustella pelaamisen muuttumisella enemmän sosiaaliseksi tapahtumaksi, oli sitten kyse Facebookin kautta pelattavista peleistä oman sijoituksen tallentavilla taulukoilla tai reaaliajassa pelattavia usean pelaajan First Person Shooter –pelejä.

Phaser ja LimeJS eivät sisällä verkkopelitukea itsessään. Phaserin plugin-järjestelmän myötä voi kuitenkin olla mahdollista kehittää oma verkkopelituki kehittämälleen pelille.

Construct 2-työkalun päivitys r168 toi verkkopelituen kaikkien käyttöön. Verkkopelituki sisältää mm. syötön viiveen vähennyksen, paikallisen ohjausmallin tallentamisen sekä tämän kautta huijauksenestön, LAN-tuen nollatasoa lähentelevällä viiveellä ja Scirran viralliset palvelimet pelaajien yhdistämiseksi samaan peliin.

Turbulenz-työkalulla tehtyjä pelejä voidaan rekisteröidyllä nimimerkillä ylläpitää verkkopeleinä Turbulenzin omilla peleille tehdyllä sivustolla, ga.me, ja tätä ennen niitä voi testauttaa joko paikallisella palvelimella tai ”Developers Hub”-palvelussa eli kehittäjän keskittimessä. Sivusto on eräänlainen web-pohjainen konsoli, joka myös tallentaa tilastoja Turbulenzilla luodun pelin tiedoista, kuten kuinka paljon pelaajia on pelannut keskimääräisesti tietyn ajan verran peliä. Tämän pohjalta ga.me voi myös markkinoida peliä laajemmalle yleisölle. On kuitenkin muistettava, että ga.me

on selainpohjainen, eikä sovellu välttämättä mobiililaitteille. Turbulenz joka tapauksessa tarjoaa verkkopelituen myös mobiililaitteille, joskin samanlaisessa muodossa kuin esimerkiksi Facebook-pelit, eli vertaillen käyttäjätunnuksen suorituksia muihin pelaajiin, eikä esimerkiksi reaaliaikaiseen yhtaikaiseen pelaamiseen kavereiden kanssa, kuten ga.me –palvelussa pyörivät pelit.

Verkkopelin puolesta voidaan siis sanoa, että Construct 2 sisältää parhaimmat mahdollisuudet verkkopelien ylläpitoon, mutta Turbulenz tukee lähinnä selainpohjaista verkkopelaamista ga.me –palvelullaan.

9.5 Isometrinen sekä 3D-tuki

Isometrinen projektio on menetelmä, jolla kolmiulotteinen kappale voidaan heijastaa kaksiulotteisena niin, että kappaleesta näkee kolme sivua. Tämä on eräänlainen tapa saada pelaaja uskomaan, että tämä pelaa kolmiulotteista peliä. 3D-tuki on tärkeä pelimoottorissa siinä vaiheessa, kun tarpeeksi kehittynyt tekijä haluaa keskittyä kolmiulotteisten pelien tekemiseen. On kuitenkin muistettava, että vaikka kolmiulotteinen tuki löytyy, on opinnäytetyönkin idea löytää sopiva pelimoottori nimenomaan aloitteleville pelinkehittäjille, joten 3D-tuki ei ole niin suuressa roolissa.

Construct 2 -työkalun yhteisön ansiosta on onnistuttu luomaan sekä isometrisiä että niin sanottuja pseudo 3D-pelejä, jotka hyödyntävät vain 2D-objekteja, eivätkä polygoneja, kuten 3D-pelit yleensä. 3D-pelejä varten kehitetään 3D-lisäyksiä, mutta niiden kehitys vie aikansa.

Turbulenz mahdollistaa sekä isometristen että kolmiulotteisten pelien kehityksen. Tämä vaatii kuitenkin jo hiukan syventävää tietoa työkalusta.

Phaser-työkalulle on tällä hetkellä työn alla isometristen pelien tuki pluginina, suurimpana esteenä on tällä hetkellä Arcade Physics -fysiikkamoottorin

muokkaaminen niin, että sillä voidaan renderöidä lisättyä ulottuvuutta. Phaserille ei kuitenkaan ole saatavilla 3D-tukea.

LimeJS ei tutkimusten mukaan sisällä isometristen pelien, eikä myöskään 3D-pelien, tukea.

Yhteenvetona voidaan siis sanoa, että mikäli haluaa jatkuvasti kehittää itseään pelinkehittäjänä, on todennäköistä, että isometristen ja 3D-pelien tekeminen tulee väistämättä vastaan. Tällöin Turbulenz tarjoaa parhaimmat työkalut tuolle siirtymälle.

9.6 Dokumentaatio ja oppaat

Jotta pelin kehittäjä pääsee työkalun kanssa mahdollisimman hyvin alkuun ja hänen on mahdollista myös palata kehitystyössä aiempiin työvaiheisiin, on dokumentaation oltava tarkkaa ja sovellettavaa työn alusta lähtien, oppaiden taas täytyy olla helposti luettavissa ja ymmärrettävissä.

Construct 2 –työkalun käyttöön on luotu oppaita jo vuodesta 2011 lähtien, ja uusia tulee säännöllisin väliajoin joko Scirran tai yhteisön toimesta. Oppaat käsittelevät kaiken fysiikasta ääniin, ja oppaita on saatavilla myös videomuodossa. Lisäksi oppaita löytyy myös omien liitännäisten luomiseen sovelluskehityspakettiin. Varsinaisten koodinpätkien dokumentaatiota ei ole saatavilla helposti, sillä osittain kaupallisena pelimoottorina ei ole eduksi, jos lähdekoodiin päästäisiin käsiksi. Tämä voi toimia kompastuskivenä Construct 2 –työkalun käyttöönotossa, sillä vaikka pelin tekeminen olisi helppoa, ei siitä ole oppijan kannalta hyötyä jos ei tiedä, miten kyseinen toiminto tapahtuu.

Turbulenz-työkalun dokumentaatiosta löytyy omiin lukuihinsa jaetut osiot, joissa opetetaan käyttöönotto sekä tekstipohjan tai selainpohjaisen käyttöliittymän kautta.

Lisäksi jokainen työkalun moduuli, kuten fysiikkamoottorit ja niiden kutsumiseen vaadittavat parametrit, on luettavissa tarkkaan.

Phaser-työkalun käyttöön on saatavilla opastusta useista eri lähteistä, eräs näistä on lessmilk.com –sivustolla, johon on koottu oppaita ja lajiteltu ne työkalun versioiden mukaan. Docs.phaser.io –sivulla voidaan katsoa aina uusimmat muutokset Phaserin viimeisimmästä versiosta.

LimeJS –työkalun dokumentaatio on saanut kehuja varsinkin käyttäjiltään, sillä paitsi että oppaat ovat selkeitä, myös dokumentaatiot luokista, joita LimeJS sisältää, ovat helposti saatavilla LimeJSn sivujen kautta. Kaikki nämä luokat on kommentoitu hyvin omilla tiedoillaan.

Dokumentaation puolesta voitaisiin siis suositella sekä LimeJS- että Turbulenz –työkaluja.

9.7 Työkalun kehittäminen sekä yhteisö

Jotta työkalu olisi mahdollisimman ajankohtainen pelinkehityksessä, sitä täytyy päivittää aktiivisesti ja hyödyntää yhteisön mahdollisia virheilmoituksia ja toiveita, että työkalu säilyisi mahdollisimman käyttäjäystävällisenä.

Construct 2 -työkalun yhteisö, joka käsittää yli 77,000 rekisteröitynyttä käyttäjää virallisilla foorumeilla, hakee jatkuvasti uusia tapoja hyödyntää pelimoottoria tai luoda uusia liitännäisiä. Tämä ilmenee muun muassa isometristen ja pseudo-kolmiulotteisten pelien luomisen mahdollistamisena. Käyttäjien toiveita kuunnellaan ja kaksiulotteiseen pelimoottoriin tuodaan parhaan osaamisen mukaan uusia ominaisuuksia. Päivittäin Construct 2:n virallisilla foorumeille tulee noin 200 uutta käyttäjää ja 350 viestiä, tällä hetkellä viestien lukumäärä on yli 350,000.

Turbulenz on saanut useita päivityksiä aina vuodesta 2010 asti, ja kehitys jatkuu yhä. Koska kyseessä on Open Source –työkalu, on myös käyttäjäkunta päässyt kehittämään sitä omalta osaltaan. Turbulenz:n käyttäjille on perustettu Google-ryhmä, jossa on tällä hetkellä 385 jäsentä. Aiheita on yli 300 ja viestejä tulee päivittäin keskimäärin noin 25 viestiä kuukaudessa. Tämä ei tietenkään kerro kaikkea, sillä virallisen ryhmän lisäksi käyttäjät ovat voineet hyödyntää muita keskustelualueita tai viestintäkeinoja pelinkehityksessä.

Phaser-työkalun versio 3.0 on jo työn alla, ja sen ”virallisilta” keskustelufoorumeilta kerätään yhteisön toiveita uuden pelimoottorin ominaisuuksiin liittyen. ”Viralliset” keskustelufoorumit sijaitsevat HTML5 Game Devs Forums –sivustolla, missä Phaser-työkalulle on oma osionsa.

LimeJS –työkalun kohtalo on käyttäjiensä käsissä, vaikka silloin tällöin virheiden korjauksia suoritetaan myös kehittäjien toimesta. Tiiviimmillään isompien päivitysten tahti on noin kuukauden välein. Julkaisustaan lähtien on työkalun päivittäminen kuitenkin tahdiltaan hidastunut huomattavasti. LimeJS –työkalun käyttäjille on myös luotu Google-ryhmä, jossa on lähempänä 400 jäsentä sekä yli 700 aihetta.

Yhteisön ja kehityksen aktiivisuuden puolesta Construct 2 on työkaluna vahva, mutta koska tarkkailussa olivat niin sanotusti viralliset keskustelufoorumit eikä niinkään kaikki internetin viittaukset työkaluun, ei voida olla täysin varmoja, mikä on tällä hetkellä vahvin. Turbulenz-työkalun tulevaisuus toisaalta näyttää yhä paremmalta, mistä kertoo muun muassa ga.me –palvelun käyttö.

9.8 Korkean vai matalan tason työkalu?

Ohjelmointirajapinta on sovellusten laatimiseen tarkoitettu matalan abstraktiotason väline, jossa käytettävät ohjelma- tai luokkakirjaston rakenteet ovat ohjelmoijan

suoraan käytettävissä. Ohjelmointirajapinnoilla viitataan tässä tapauksessa pelinkehityksen työkaluihin. Näitä on kaksi tasoa, matala ja korkea. Matalan tason työkalu vaatii paljon koodaamista, mutta on erittäin joustava. Korkean tason työkalu onkin sitten tämän vastakohta, eli heikko joustavuus, mutta vähemmän koodaamista.

Construct 2 on pääosin korkean tason työkalu, se sisältää tarpeelliset pohjat valittujen peligenrejen luomiseen sekä vaatii erittäin vähän koodaamista, ellei itse halua koodata esimerkiksi liitännäisiä, vaikka näidenkin käyttömahdollisuudet ovat rajallisia.

Turbulenz tarjoaa sekä korkean että matalan tason työkalut, mikä on erikoista varsinkin Open Source –työkalussa. Korkeatasoisten työkalujen käytössä suositellaan tietenkin dokumentoitujen objektien, funktioiden ja asetusten käyttöä sovellusta luodessa.

Phaser on korkean tason työkalu, mutta siihen on sisällytetty oma lisäysten luonti- ja hallintajärjestelmä, jolla voidaan luoda omia lisäyksiä Phaser-työkaluun ja jakaa niitä muille sen käyttäjille.

Myös LimeJS on korkean tason työkalu, se sisältää tarpeelliset työkalut muutaman erilaisen genren luomiseen, mutta lisäysten tekeminen on mahdollista kohtuuden rajoissa.

Työkalujen tason suhteen on Turbulenz kaikkein joustavin, sillä se tarjoaa molempien tasojen työkalut, jotta voi valita, haluaako tehdä mahdollisimman paljon alusta loppuun vai valita jonkin tietyn pelityypin, josta tehdä pelin.

9.9 Renderöintinopeus

Renderöintinopeudella tarkoitetaan sitä, kuinka nopeasti peli muuntaa digitaalisen tiedon näytölle sopivaan esitysmuotoon. Tähän vaikuttaa monta eri tekijää, ja jokaisella työkalulla on omat tapansa nopeuttaa renderöintiä mahdollisimman pienellä vaivalla.

Kriteerinä renderöintinopeus on vaikea määrittää, sillä vaikka saman pelin testaisi eri pelimoottoreilla joillakin valituilla laitteilla, on renderöintinopeus niin useasta tekijästä kiinni, että yhtä tiettyä lukua tietyn pelimoottorin renderöintinopeudesta on lähestulkoon mahdotonta. On suositeltavaa, että pelit tehdään alun alkaen mobiilialustoja ajatellen, ja hyödynnetään eri pelimoottoreiden omia työkaluja nopeamman renderöinnin saavuttamiseksi.

Scirra antaa muutaman vinkin joilla parantaa sillä rakennetun pelin suorituskykyä vähemmän tehokkailla laitteilla:

1. Salli WebGL. Laitteet, jotka tukevat WebGL-ohjelmointirajapintaa, parantavat pelin suorituskykyä.
2. Vältä liian monen objektin tai partikkeli-efektin käyttöä.
3. Aseta samanlaiset efektit samalle tasolle. Efektien välillä vaihteleminen voi vähentää suorituskykyä.
4. Vältä objekteja, joissa on suuria läpinäkyviä alueita. Lisäksi, leikkaa kuvista mahdollisimman paljon läpinäkyvää pintaa. Voit myös kokeilla suurempien kuvien leikkaamista pienempiin osiin.
5. Vältä objektien päällekkäisyyttä tai liian usean objektin kosketusta toisiinsa. Päällekkäisten alueiden pikselit on renderöitävä uudelleen ja uudelleen, tämä heikentää suorituskykyä erittäin nopeasti. Törmäyksissä täytyy laskea useita eri tapauksia, jotka voivat yhteentörmäyksestä tulla jo yhdessä sekunnissa.

Lisäksi Scirran oppaassa listataan yleisiä syitä huonoon suorituskykyyn.

- Ei laitteiston kiihdytystä. Jopa hidas Canvas 2D –renderöinti saa suorittamiseen apua grafiikkaprosessorilta. Joidenkin laitteiden grafiikka-ajurit eivät kuitenkaan toimi tasaisesti, ja selain voi sulkea grafiikkaprosessorin käytöstä kaatumisen välttämiseksi. Vanhemmissa laitteissa grafiikkaprosessorikiihdytys ei ole yksinkertaisesti tuettu.
- Liian monta fysiikkaobjektia. Fysiikan mallintaminen vaatii paljon prosessorin tehoja. Mielummin muutama suurempi fysiikkaobjekti kuin monta pientä sellaista.
- Aivan liian monta objektia, efektiä ja/tai partikkeliefektiä. Varsinkin turhien efektien käyttö voi haitata huomattavasti suorituskykyä.

Boris Smus kertoo 29. lokakuuta 2013 muokatussa artikkelissaan yleisesti HTML5-piirtoalueen suorituskyvyn parantamisesta. Artikkelissa suositellaan muun muassa etukäteen objektien piirtämistä senhetkisen ruudun ulkopuolella erillisellä piirtoalueella, jonka jälkeen ne tuodaan senhetkiselle ruudulle. Erittäin tehokasta olisi myös piirtää ruudulle kokonaan uuden tilan sijasta vain muutokset verratessa edelliseen tilaan.

Turbulenz-työkalu sisältää oman canvas- eli piirtoalue-objektin, jota suositellaan käyttämään selainten omien piirtoalue-objektien sijasta juuri grafiikkaprosessorin hyödyntämiseksi, sillä vaikka nykyisissä selaimissa käytetään grafiikkaprosessoria suorituskyvyn parantamiseksi, on silti Turbulenz-työkalun oma piirtoalue-objekti ollut testeissä suorituskyvyltään tehokkaampi.

Muissa työkaluissa ei ole tavallisesta HTML5-piirtoalueen renderöinnistä poikkeavaa menetelmää ei ole dokumentointuna.

9.10 Yhteenveto ja suositus

Kaiken kaikkiaan jokaisella työkalulla on omat etunsa pelinkehityksessä, mutta tämän opinnäytetyön kannalta tärkeintä on vastata kysymykseen: Minkä pelinkehityksen työkalun oppiminen edistää oppimistani paitsi pelin-, myös sovelluskehityksen oppijana?

Kaikkia edellä mainittuja tekijöitä tutkien ja vertaillen toisiinsa, olisi oppimisen kannalta paras vaihtoehto opetella käyttämään työkalua, joka hyödyntää yleisesti sovelluskehityksessä olevaa koodikieltä, kuten JavaScript. Lisäksi tämän työkalun olisi hyvä olla mahdollisimman joustava, jotta sillä voidaan toteuttaa oppijan kokemuksen ja kiinnostuksen mukaisesti yksinkertaisia pelejä tai jopa isompia projekteja. Kolmiulotteisten pelien tukeminen on myös plussaa, mikäli opiskelija haluaa suuntautua suurempaankin peliteollisuuteen töihin. Lisäksi laaja yhteisö sekä mahdolliset tuki- ja jakelukanavat voivat tukea työkalun käyttöä opiskelujen ulkopuolella.

Näihin kriteereihin sopisi parhaiten Turbulenz, joka sisältää kaiken edellä mainitun. Dokumentointi ja oppaat ovat selkeitä, työkalua päivitetään tasaiseen tahtiin, se sisältää sekä korkean että matalan tason työkalut riippuen projektin vaatimasta työstä, siihen voidaan sisällyttää omia liitännäisiä ja voi halutessaan hyödyntää visuaalista käyttöliittymää paikallisella palvelimella.

Mikäli visuaalinen käyttöliittymä olisi tärkein kriteeri oppimisen sijasta, Construct 2 työkaluna olisi hyvä vaihtoehto Turbulenz-työkalun sijasta, mutta koska Construct 2 vaatii oppijalta vähemmän koodaamista kuin Turbulenz, se ei tukisi oppimista yhtä hyvin. LimeJS sekä Phaser ovat myös paljolti koodipohjaisia ja tukisivat oppimista, mutta niiden joustavuus on selkeästi heikompi kuin Turbulenz-työkalulla. Ne eivät esimerkiksi tarjoa 3D-tukea.

10 Pohdinta

Tutkimuksessa pyrittiin siihen, että saataisiin mahdollisimman erilaisia työkaluja esiteltäväksi mahdollisina vaihtoehtoina JAMKin käyttöön pelinkehityksen opettamisessa. Tietenkin tämä täytyi ottaa opiskelijankin kannalta huomioon, sillä opiskelijalla täytyy olla mahdollisuus saada käyttää ko. työkalua ilman erillistä maksullista lisenssiä, jotta sen käytön opiskelu olisi mahdollista opetuksen ulkopuolella. Opinnäytetyön tietoja voisi pitää lähinnä esittelynä kyseessä olevien työkalujen käyttöön eikä niinkään perustavina oppaina niistä. Näin jätettäisiin myös tilaa mahdolliselle jatkotutkimukselle, joka voisi käsittää vielä useampia työkaluja, kunhan ne vielä kehittyvät.

Opinnäytetyön edetessä ilmeni erittäin tärkeä seikka, joka täytyi ottaa huomioon tutkimusta tehtäessä: työkalun merkitys opiskelija oppimisen kannalta. Oli kysyttävä, mikä työkalu tukisi opiskelijan oppimista paitsi pelinkehittäjänä, myös sovelluskehittäjänä. Tällöin täytyi ottaa huomioon myös muun muassa koodikieli, jota suositeltava pelinkehityksen työkalu käyttää.

Teorian ja tutkimustulosten yhdistäminen osoittautui haasteeksi. Teoria kertoi Suomen pelinkehityksen historiasta ja todistaa, että voidakseen vastata alati muuttuvaan ja kasvavaan pelialaan, on Suomen koulutettava osaavia pelinkehittäjiä. Tämän täytyy näkyä myös opetuksessa, mutta pelkän pelinkehityksen opettamisella ei ole oppimisen kannalta suurta hyötyä. On myös osattava yleistä sovelluskehitystä, jotta työllistyminen tietotekniikan alalla ei rajoitu vain rajattuun toimialaan, tässä tapauksessa pelinkehitykseen. Pelien koodaaminen kuitenkin perustuu usein johonkin tunnettuun koodikieleen, kuten JavaScript:een tai C++:aan, ja loppujen lopuksi kyse on vain eräänlaisesta sovelluskehityksestä.

Lähteet

Good, O. 2009. Ubisoft: All Our Games Will Do This UPlay Thing. Kotaku (2009). Viitattu 17.3.2014. http://www.kotaku.com.au/2009/11/ubisoft-all-our-games-will-do-this-uplay-thing/

Erin Catto – About Box2D. Viitattu 12.11.2014 http://box2d.org/about/

Hiltunen, K.-P., Latva, S. & Kaleva, J-P. 2013. Peliteollisuus – kehityspolku. Viitattu 21.2.2014. http://www.tekes.fi/Julkaisut/peliteollisuus_kehityspolku.pdf

Introduction to Assembly. (2007) Viitattu 21.2.2014. http://www.assembly.org/summer07/asm/introduction

Leigh, A. N.d. Electronic Arts Closes EA Link Service, Unveils EA Store. Viitattu 22.2.2014. http://www.gamasutra.com/php-bin/news_index.php?story=15610

Lunden, I. 2012. Windows Phone Is Taking Share From RIM, But It's Still Nowhere Near Breaking Through The Android/iOS Stronghold: Research (2012). Viitattu 23.4.2014. http://techcrunch.com/2012/10/01/windows-phone-is-taking-share-from-rim-but-its-still-nowhere-near-breaking-through-the-androidios-stronghold-research/

McCracken, H. 2013. Who's Winning, iOS or Android? All the Numbers, All in One Place. Viitattu 22.4.2014. http://techland.time.com/2013/04/16/ios-vs-android/

Mäyrä. F. & Ermi, L. 2013. Pelaajabarometri 2013 – Mobiilipelaamisen nousu. Viitattu 9.5.2014. http://tampub.uta.fi/bitstream/handle/10024/95150/pelaajabarometri_2013.pdf?sequence=1

Pelialalle! N.d. Jyväskylän Yliopisto informaatioteknologian tiedekunta: Uranäkymiä. Viitattu 8.5.2014. http://www.itjkl.fi/uranakymia-2/#pelialalle

Performance Tips. N.d. Viitattu 16.11.2014. https://www.scirra.com/manual/134/performance-tips

Siliconera. N.d. Nintendo Are "Looking At" Bringing Unity To Nintendo 3DS (2014). Viitattu 24.3.2014. http://www.siliconera.com/2014/03/20/nintendo-looking-bringing-unity-nintendo-3ds/

Suomen Pelinkehittäjät Ry. Suomen Pelialan Strategia 2010–2015. Visio 2020. Viitattu 21.2.2014. http://www.neogames.fi/wp-content/uploads/2013/05/Pelistrategia-2010-2015.pdf

Turbulenz documentation. N.d. Viitattu 16.11.2014. http://docs.turbulenz.com/index.html

Unity – Fast Facts. Viitattu 20.1.2014. http://unity3d.com/company/public-relations

Liitteet

Liite 1: Vertailutaulukko

	Mobiili- ja tablettituki	**Joustava käyttöliittymä**
Construct 2	Construct 2 tukee kosketusnäytöllisiä laitteita, sillä ohjauksen kääntäminen kosketusnäytöllisille alustoille on yksinkertaista.	Käyttöliittymän joustavuus riippuu luotavan pelin laadusta, mutta pelistä löytyy valmiit pohjat mm. tasoloikalle, ylhäältä kuvattaville ammuskeluille, fysiikka-aivopähkinöille, reaaliaikaisille strategioille ja vertikaalisille avaruus-ammuskeluille.
Turbulenz	Turbulenz tekee mahdolliseksi pelien kehittämisen kosketusnäytöllisille laitteille. Tämän mahdollistaa HTML5-tuki sekä erillinen tapa työstää pelit erillisinä sovelluksina nimenomaan mobiililaitteille.	Turbulenz vaatii alleen pelkästään tekstieditorin ja selaimen, kuten Google Chrome tai Mozilla Firefox.
Phaser	Phaser on rakennettu nimenomaan mobiiliselaimia ajatellen, myös erillisten mobiilisovellusten kehitys mahdollista. Lisäksi Phaser omaa skaalaus-koodin, joka voi skaalata pelin minkä kokoiselle näytölle tahansa.	Phaseriä voi muokata MightyEditorilla, joka on selainpohjainen. Sen avulla voi luoda yksinkertaisia 2d-pelejä ja myös päästä käsiksi Phaserin lähdekoodiin. Phaseriä voidaan käyttää myös muun pelinkoodauksen ohessa.
LimeJS	LimeJS on suunniteltu hyödynnettäväksi kosketusnäytöllisillä mobiililaitteilla, tietenkin myös selainpelien kehitys on mahdollista.	Sisältää oman Python-työkalun, jolla voidaan luoda ja muokata omia LimeJS-projekteja, kuten JavaScript-koodeja.

	Integroitu fysiikkamoottori	**verkkopelituki**
Construct 2	Löytyy, mutta Scirra itse suosittelee oppaissaan, että enimmäismäärä samanaikaisesti näkyviä fysiikkaobjekteja on mahdollisimman vähäinen, koska fysiikka-objektien renderöinti kysyy prosessointitehoja, joka varsinkin mobiililaitteilla on pieni tietokoneisiin nähden.	päivitys r168 tuo verkkopelituen kaikkien käytettäväksi. Tämä sisältää muun muassa tehostetun syötön viiveen vähennyksen, sisäänrakennetun paikallisen ohjausmallin tallentamisen, joka estää myös huijauksen, LAN-tuen lähes nollatason viiveellä, binääridatasiirron osittaisen ohjauksen kaistan tarpeen minimoimiseksi ja Scirran viralliset palvelimet pelaajien yhdistämiseksi.
Turbulenz	Sekä kaksi- että kolmiulotteisiin peleihin soveltuvat fysiikkamoottorit saatavilla.	Kommunikointikanavat TCP-soketin kautta, hyödyntäen selaimen WebSocket-tukea. Pystyy kommunikoimaan HTTP-palvelimien kanssa, jotta voidaan jakaa tavalliset HTTP- sekä HTTPS-portit WebSocket-palvelimella.
Phaser	Phaser sisältää useita fysiikkamoottoreita, joista jokaisella on oma käyttötarkoitus. Arcade Physics käsittää nopean AABB-kollision mallinnuksen, Ninja Physics antaa mahdollisuuden käyttää monimutkaisempia muotoja ja mäkiä, jolloin se soveltuu maisemiin. P2.JS on objekteille sopiva fysiikkamoottori, sisältäen polygonituen, jouset jne.	Ei ole.
LimeJS	Sisältää Box2D-kirjaston.	Ei ole.

	isometrinen tuki	**3D-tuki**
Construct 2	Toimivan yhteisön ansiosta on selvitetty, että Construct 2:lla voidaan luoda isometrisiä pelejä.	Yhteisön toiminnan ansiosta voidaan luoda ns. pseudo 3D-pelejä, jotka hyödyntävät 2D-objekteja, mutteivät polygoneja. Mahdollisuuksia voi löytyä 3D-pluginejen myötä, mutta niiden tekeminen vie aikaa.
Turbulenz	Mahdollista, vaatii edistynyttä tuntemusta pelimoottorista sekä muista hyödynnettävistä työkaluista.	Saatavilla sekä matalan tason työkaluun että korkean tason työkaluun.
Phaser	Tällä hetkellä työn alla, saatavana pluginina. Suurinpana työpanoksena nähdään Arcade Physicsin muokkaaminen siten, että sillä voidaan renderöidä lisättyä ulottuvuutta.	Ei ole.
LimeJS	Ei ole varsinaista	Ei ole.

	Dokumentaation ja oppaiden laatu
Construct 2	Oppaita on luotu vuodesta 2011 lähtien, ja uusia tulee aina säännöllisin väliajoin joko itse kehittäjien tai yhteisön toimesta. Oppaiden vaihtelevuus on laaja, aina fysiikkamoottorin käytöstä ääniin ja video-oppaisiin. Dokumentaatio sisältää ns. virallisen opaskirjan lisäksi oppaan omien liitännäisten luomiseen sovelluskehityspaketin (SDK, Software Development Kit) myötä.
Turbulenz	Virallinen dokumentaatio sisältää mm. opastuksen perustoimintoihin sekä harjoitussovelluksen, listaa ja esittelee pelimoottorin ominaisuuksia sekä tiedossa olevat ongelmat sekä niiden mahdolliset ratkaisut.
Phaser	Erittäin laajat opastukset saatavilla useista eri lähteistä. Erinäisiä oppaita on koottu http://www.lessmilk.com/phaser-tutorial/ ja ne on listattu Phaserin versioiden mukaan. Lisäksi uusien päivitysten ilmestyessä kootaan docs.phaser.io -osoitteeseen aina uuden päivityksen selkeimmät muutokset.
LimeJS	Käyttäjiensäkin kehuma dokumentaatio on erittäin selkeä, aina oppaista yksityiskohtaiseen dokumentaatioon luokista ja tiedostoista, joita LimeJS sisältää. Omat oppaat sekä dokumentaatiot työskentelyn aloittamiseen on saatavilla, lisäksi koodi itsessään on hyvin kommentoitu tarpeellisilla tiedoilla (kuten mitä mikäkin koodi tekee).

	Työkalun kehityksen aktiivisuus	**Yhteisön koko**
Construct 2	Yhteisö hakee jatkuvasti uusia tapoja hyödyntää pelimoottoria, kuten esimerkiksi isometristen sekä pseudo-kolmiulotteisten pelien luomisessa. Kehittäjät ottavat myös huomioon käyttäjien toiveet ja parhaansa mukaan tuovat uusia ominaisuuksia kaksiulotteiseen pelimoottoriin.	Yli 77,000 rekisteröitynyttä käyttäjää foorumeilla, keskimäärin 200 uutta foorumikäyttäjää päivittäin, yli 350,000 foorumiviestiä, keskimäärin noin 350 foorumiviestiä päivittäin.
Turbulenz	Pelimoottoria on kehitetty vuodesta 2010 tähän päivään asti ja kehitetään yhä, koska kyseessä on Open Source -moottori, jota voi myös itse muokata, kehittyy työkalu myös yhteisön osalta.	Turbulenz Engine Users Forum Googlen Ryhmät-palvelussa, 319 aihetta, 373 ryhmän jäsentä. Keskimäärin 60 viestiä kuukaudessa viimeisen 11 kuukauden sisään (7.10.2014)
Phaser	Phaser 3.0 on jo työn alla, ja siihen kerätään yhteisöltä toiveita erinäisiin ominaisuuksiin liittyen. Kaikkea ei tietenkään voida toteuttaa, mutta tietenkin osaava yhteisö osaa toivoa pelimoottorin kestävyyden mitoissa asioita.	Sivuilta saatavilla oleva linkki vie HTML 5 Game Devs foorumeille (www.html5gamedevs.com/forum), joka sisältää oman osion Phazerille sekä sillä tehdyille peleille.
LimeJS	Tällä hetkellä lähinnä yhteisön käsissä, vaikka muutamia bugien korjauksia julkaistaan aina tarpeen tullen, kovimmalla tahdilla noin kuukauden välein. Päivitystahti on selkeästi laskenut alkutaipaleen jälkeen.	Google Groups, ryhmässä 753 aihetta, 394 jäsentä.

	Matalan tason työkalu vai korkean tason työkalu?
Construct 2	Construct 2 on korkean tason työkalu, sillä se tarjoaa hyvin paljon omia välineitään käyttöön valmiiksi joustavuuden hinnalla. Tietenkin tähän työkaluun voidaan itse koodata liitännäisiä, jolla voi saada aikaiseksi sellaisia pelejä kuin haluaa tehdä, rajoituksella että ne toimivat kaksiulotteisesti tai enintään pseudo-kolmiulotteisesti, ja näiden liitännäisten luomiseen ja muokkaamiseen on saatavilla oppaita.
Turbulenz	Turbulenz tarjoaa sekä korkean että matalan tason työkalut, joihin voi tarttua mielensä mukaan. Korkeatasoisten työkalujen käytössä suositellaan vain dokumentoitujen objektien, funktioiden ja asetusten käyttöä sovellusta luodessa. Varsinkin matalan tason työkalut eivät rajoitu mihinkään pelityyppeihin.
Phaser	Korkean tason työkalu. Voidaan sisällyttää omia liitännäisiä.
LimeJS	Korkean tason työkalu

	Renderöintinopeus
Construct 2	Renderöintinopeuden maksimoimiseksi Scirra opastaa välttämään liian monien objektien ja hiukkasefektien käyttöä, asettamaan samanlaisia efektejä omaavien objektien asettelua samalle tasolle, leikkaamaan kaikki kuvat niin, että ne sisältäisivät mahdollisimman vähän läpinäkyvää pintaa, objektien limittäytymisen määrän vähentämistä,välttämään liiallista taustan renderöintiä uudelleen ja uudelleen sekä testaamaan alun alkaenkin mobiililaitteilla.
Turbulenz	On täysin kehittäjästä kiinni, kuinka raskaita pelejä tahtoo tehdä, lähtökohtaisesti mobiililaitteilla pelien testaaminen on suositeltavaa ja että pelit kehitettäisiin alun alkaenkin mobiililaitteille. Liian suuret pelit voivat hidastella varsinkin vanhemmilla älypuhelimilla.
Phaser	Phaser on rakennettu nimenomaan mobiiliselaimia ajatellen, mutta on kyse siitä, miten raskaan pelin Phaserilla kehittää.
LimeJS	LimeJS sisältää oman redneröijänsä, joka hyödyntää laitekohtaisesti eri elementtejä, joilla renderöidä nopeammin pelejä. Esimerkiksi Applen iOS versio 4.2 on hidas käyttäessään canvas-elementtiä, mutta käyttää sen sijaan GPU -nopeutettua CSS-elementtiä. Tavalliset PCt hyödyntävät taas Canvas-elementtiä nopeammin. lime.Renderer pystyy kääntämään tämän yhdellä rivillä koodia, joko DOM tai CANVAS -formaatissa. Tulevaisuudessa aiotaan hyödyntää myös WebGL-renderöintiä.

	Työkalun hinta
Construct 2	Henkilökohtainen lisenssi 99.99€, yrityslisenssi 329.99€, opiskelijalisenssi oppilaitoksen koneille 21.99€/kuukausi, 194.99€/vuosi. "Take home license" eli opiskelijan koululta kotikäyttöön saatava lisenssi on 1.19€/oppilas/kuukausi, 12.99€/oppilas/vuosi.
Turbulenz	Ilmainen
Phaser	Vapaasti käytettävissä. "Discover Phaser" -opas saatavilla kirjana 29$, oppaan sisältämät lähdekoodit $59 sekä kolme valmista peliä $129.
LimeJS	Ilmainen.