**Yleistä kurssista**

* Tavoite: Opiskelija osaa suunnitella, dokumentoida, toteuttaa ja testata 2D-pelin C#-ohjelmointikielellä.
* Käytetään Unity- tai MonoGame-pelimoottoria, myös muut ympäristöt ovat perustellusta syystä mahdollisia.
* Kurssi suoritetaan toteuttamalla 2d-peli 1-3 hengen ryhmissä.
* Koodien hallinnassa käytettävä versionhallintaa. <https://repo.kamit.fi/>
* **Jokaisen on osallistuttava ohjelmointiin ja pidettävä projektipäiväkirjaa josta oma panos käy ilmi. Lisäksi lähdekoodien alkuun on merkittävä selvästi kuka ne on luonut ja kuka niitä on milloinkin muokannut (muokkaushistoria).**
* Kurssin lopussa jokainen täyttää itsearvioinnin, jossa on mukana myös arviointi muusta ryhmästä.

**Arviointikriterit**

* Kiitettävä (5):

Opiskelija osaa itsenäisesti soveltaa taitoja ja hyödyntää monipuolisesti tietoa ongelman ratkaisussa.

* Hyvä (3-4):

Opiskelija osaa valita peliprojektiin sopivia, tarkoituksenmukaisia toimintatapoja hankkimansa tiedon ja ohjeistuksen perusteella.

* Tyydyttävä (1-2):

Opiskelija osaa toimia ohjattuna asianmukaisesti, noudattaen annettuja sääntöjä ja ohjeita. Opiskelija käyttää toiminnassaan tarkoituksenmukaisesti pelialan tekniikoita ja malleja.

**Itsearviointi**

* Itsearviointilomake <https://edu.kamit.fi/pluginfile.php/241794/course/section/52799/Itsearviointi.doc>. **Lisää lomakkeen loppuun arvio myös muiden ryhmän jäsenten panoksesta peliprojektissa**.
* Täytä lomake ja palauta se projektipäiväkirjan kanssa palautuslaatikkoon viimeistään 11.3.2018.
* Pakkaa lomake ja projektipäiväkirja yhdeksi zip-paketiksi. Jos projektipäiväkirjaan pääsee www:n kautta, niin lisää maininta tästä lomakkeeseen, jolloin erillistä päiväkirjatiedostoa ei tarvita.
* Yksi henkilö/ryhmä palauttaa pelin koodit alla oleviin palautuslaatikoihin. Voit vaihtoehtoisesti palauttaa linkin repositoryyn.

https://edu.kamit.fi/theme/image.php/clean/core/1527165348/spacerUNITY

* Kaupallinen ohjelmisto, useita lisensointivaihtoehtoja.
* Kehitys: Windows, Linux, OS X
* Kohde: "Riittävästi"
* C#, Boo ja JavaScript

**Unity-tutoriaaleja**

1. Ensimmäinen tutoriaali. Jos et ole ennen ohjelmoinut Unityllä, aloita tästä: <https://unity3d.com/learn/tutorials/projects/roll-ball-tutorial>
2. Toinen tutoriaali, yksinkertainen räiskintä: <https://unity3d.com/learn/tutorials/projects/space-shooter-tutorial>
3. Isometrinen 3d-peli: <https://unity3d.com/learn/tutorials/projects/survival-shooter-tutorial>
4. Kaksinpeli yhdellä näppäimistöllä, pelimekaniikan teko ym.: <https://unity3d.com/learn/tutorials/projects/tanks-tutorial>
5. Proseduraalinen sisällön generointi: <https://unity3d.com/learn/tutorials/projects/2d-roguelike-tutorial>
6. Proseduraalinen sisällön generointi jatkuu: <https://unity3d.com/learn/tutorials/projects/procedural-cave-generation-tutorial>
7. 2D-pelin kehittäminen: <https://unity3d.com/learn/tutorials/projects/2d-ufo-tutorial>
8. Sekalaisia pelimekaniikkoja ym. :[https://unity3d.com/learn/tutorials/lets-try](https://unity3d.com/learn/tutorials/lets-try" \t "_blank)

C# tutoriaali: <https://www.tutorialspoint.com/csharp/>   
Unity manuaali: <https://docs.unity3d.com/Manual/index.html>

Peliohjelmointi 1 PROJEKTIPÄIVÄKIRJA

Maanantai 12.11.2018

Aloitan kurssin miettimällä tehtävänannon mukaista pelitoteutusta. Olen rajannut kehityksen 2D Unity-peliin, joka käyttää C# -olio-ohjelmointikieltä pohjanaan.

Pelin tulisi olla yksinkertainen, mutta olla helposti laajennettavissa sovellettavalla sisällöllä. Ensimmäisinä ideoina herää perinteinen platformer -tai shmup-peli, joista on tarpeen helppo löytää kehitystä tukevaa materiaalia.

Alustavasti kallistun enemmän shumpin suuntaan, sillä uskon platformer-pelin olevan graafisesti vaativampi toteutus ja haluan pitää keskittymiseni ohjelmoinnin puolella. Grafiikkaa on tietysti helppo lisätä jälkeenpäin ajasta riippuen. Perehdyn mahdollisuuksiini tutkimalla työmetodeja netistä ja listaan ajatuksia, sekä pelimekaniikkoja, joita haluan projektiltani. Selvitän loppupäivän aikana, kumpaan peliprojektia lähden työstämään.

Tiistai 13.11.2018

Päätin ottaa työn alle Shmup-pelin. Pelikehitystä varten lajityypistä löytyy paljon viiteresursseja, mikä vähentää kehityksen aikana syntyviä mahdollisia ongelmia. Peli ei myöskään ole graafisesti yhtä vaativa, kuin platformer-peli.

Muutoksena moniin Unitylla toteutettuihin shmup-peleihin, haluan tehdä pelin kulkemaan horisontaalisesti. Aloitan kehityksen muutamalla nopealla placeholder-grafiikalla ja järjestän githubin versiohallintaa varten.

Keskiviikko 14.11.2018

Työstin pelaaja-aluksen, vihollisaluksen ja ammusgrafiikat. Toistaiseksi päätän jättää grafiikan teon myöhempään vaiheeseen. Huomasin keskittyväni liikaa projektin visuaaliseen puoleen graafikkona. Minun pitää pysyä tehtävänannon tavoitteissa ja perehtyä ohjelmointikieleen.

Lauantai 22.12.2018

Projekti on ollut toistaiseksi jäissä. Muut projektini ovat vaatineet huomiotani. Integroin Visual Studion Unityyn ja aloitin ohjelmoinnin pelaaja-aluksen liikkumisella, Unityn Rigidbody-komponentti tuottaa minulle päänvaivaa. Liikkumisen säätäminen oikeaksi on haastavaa. Pelaaja-alus on liikkumiseltaan liian liukas ja vaikeasti hallittava. Shoot Em up -peleissä liikkumisen tarkkuus on tärkeää. Tämä saattaa tuottaa ongelmia myöhemmin kehityksessä.

5.1.2019

Aloitin projektin työstämisen Shoot Em Up -pelinä, mutta päädyin hylkäämään ajatuksen platformer-pelin puolesta.

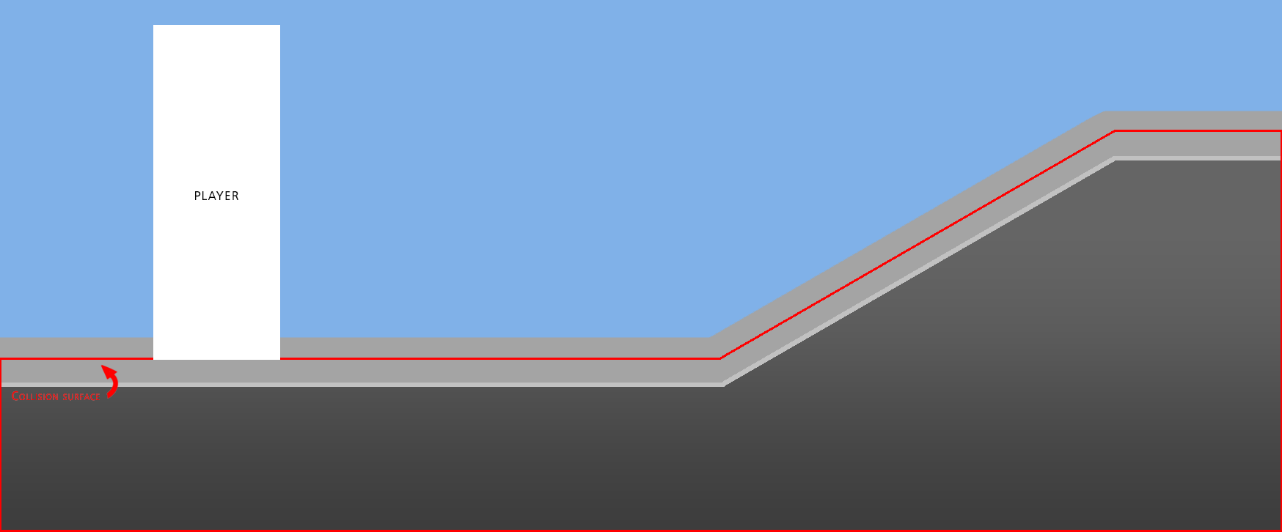
Päätin muuttaa projektin platformer-peliksi, koska sen tutkiminen osoittautui itselleni hyödyllisemmäksi vaihtoehdoksi. Voin käyttää projektissa opittuja periaatteita helpommin muissa projekteissa. Huomasin myös, että 2D-platformer-peleille on tarjolla laadukkaampaa viitemateriaalia, jota on helpompi soveltaa pelikehityksessä. Päätin myös keskittyä enemmän peliohjelmointiin ja pitää grafiikan toistaiseksi yksinkertaisena. Saatan perehtyä 2D-animaatioiden integraatioon myöhemmin.

Aloitin liikkeen tekemisen rigidbodylla, mutta törmäsin ongelmiin. Päätin ottaa metodiksi 2D-peleissä tutummin käytetyn liikkumismuodon. Unity-integroiminen tuottaa itselleni välillä vaikeuksia C#-kieleen perehtymisessä. En aina tiedä mikä toiminto viittaa mihinkin pelimoottorin puolella

Pelaajan liikkumisen pohjana päätin käytää [Sebastian Laguen](https://www.youtube.com/playlist?list=PLFt_AvWsXl0f0hqURlhyIoAabKPgRsqjz) julkaisemaa videomateriaalia 2D-ohjaimen luomisesta. Videot perehtyvät selittämään ohjelmoinnissa käytettävän logiikan ja matemaattiset konseptit tavallista selkeämmin. Metodit ovat tähän mennessä minulle ennestään tuttuja Unityn ulkopuolella. Jätin pois koodista kääntymisessä tapahtuvan hidastumisen liikkeiden tarkkuuden säilyttämiseksi.

7.1.2019

Toteutin projektiin liikkumisen kallistuneilla alustoilla (ylä- ja alamäet). en ole aikaisemmin toteuttanut raycast-törmäystunnistusta vinoille pinnoille. Koin aiheen omalta kohdalta mielenkiintoiseksi ja toivon voivani tulevaisuudessa soveltaa samankaltaista menettelyä tulevissa 2D-projekteissa. Pelaaja hahmo liikkuu kallistuneilla pinnoilla täysin suorana, mikä näyttää täydellisestä ortografisesta näkökulmasta huonolta. Tämä voidaan kuitenkin korjata antamalla käveltäville pinnoille kevyt perspektiivi laskemalla törmäystunnistusta pinnan sisälle ja piirtää alustan sivuprofiili törmäystunnistusrajan sisään.



14.1.2019

Tein projektiin liikkuvat alustat. En ollut aiemmin perehtynyt liikkuvien alustojen ja pelaajan törmäystunnistukseen näin syvällisesti. Osuus herätti paljon huomioon otettavia seikkoja. Alustat ovat helposti säädettävissä olevia. Ne voivat liikkua halutuilla nopeuksilla ja syklisillä radoilla, jotka voidaan määrittää suoraan Unityn puolella. Pidän alustojen koodiratkaisun mielessä tulevia projekteja varten.

15.1.2019

Seinähyppy on yleisesti toivottu ominaisuus platformer-peleissä. Se avaa paljon mahdollisuuksia kenttäsuunnittelussa.

Seinähyppy, hyppykontrolli, läpiliikuttavat alustalta.

Pudottautuminen alustalta voisi olla perinteisempi (alas + hyppy).

Kamera. 2D peleistä tuttu toteutus. Koodi alkaa mennä ymmärtämisen ylitse. Unity hämmentää asioita entisestään.

Korjauksia.

16.1.2019

Sain 2D-ohjaimen valmiiksi. Ohjaimessa on pieniä asioita, joita mietin korjaavani. Yksi on kameran keskittyminen pelaajaan pelin alkaessa, mikä on liian hyppäävä. Haluisin kameran pysyvän paikallaan keskittyen pelaajaan. Toinen on alustojen läpi pudottautumiseen käytetty koodi. Pudottautuminen on ajastettu törmäystunnistuksen toimivuuden kannalta, jotta pelaaja ei jää jumiin alustaan. Ajastus on kuitenkin mielestäni liian pitkä, sillä pelaaja voi pudottautua kerralla useamman alustan läpi. Pudottautumisessa pitäisi olla enemmän hallittavuutta, joka nojaa enemmän pelaajan syötteeseen.

Tämän hetkinen tavoitteeni on lisätä projektiin soveltavaa koodia käyttäen 2D-ohjainta liikkumisratkaisuna.

Tilesets, Shooting, dash.

17.1.2019

Liikkuminen toimii, mutta minulla on vaikeuksia integroida siihen muita soveltavia pelikonsepteja. Aloittelijana Unityyn ja C#-ohjelmointikieleen, pohjustukseni on liian erilainen muihin verkkomateriaaleihin verrattuna voidakseni viitata niihin ja lisätä ajattelemiani lisäominaisuuksia ilman ongelmia. Mikään toinen koodirakenne ei jäsenny samanlaisesti toteuttamani pohjan kanssa. Jos osaisin ennestään Unity C#-ohjelmointia, saattaisin pystyä tulkitsemaan niiden eroja paremmin ja tehdä tarvittavat muutokset koodien yhteensovittamiseksi.