Python Y2 projektityö – Tornipuolustus peli

Noah Nettey 474788

EST vk.2

6.5.2016

Yleiskuvaus

Kyseessä on siis tornipuolustus peli jossa viholliset lähtevät tietystä ennalta määrätystä pisteestä ja kulkevat ennalta määrätyn reitin läpi yrittäen päästä maaliin. Reitin varrelle pelaaja asettaa torneja, jotka ampuvat vihollisia, joilla hänen on tarkoitus estää vihollisten pääsyy maaliin. Jos liian monta vihollista pääsee maaliin, peli on hävitty. Viholliset tulevat aalloittain ja aaltojen välissä pelaaja voi asettaa torneja mielensä ja raha tilanteensa mukaan. Jokaisesta tapetusta vihollisesta saa tietyn verran rahaa palkkioksi, riippuen vihollisen vaikeudesta. Vihollisille on mahdollista määrittää nopeus ja kuinka paljon ne kestävät osumaa, muuten vihollisilla ei ole keskenään eroja. Torneja on kahden tyyppisiä ns. fiksuja ja tyhmiä torneja. Tyhmät tornit ampuvat aina lähimpänä olevaa vihollista. Fiksut tornit ottavat kohteekseen vihollisen ja ampuvat sitä niin kauan kunnes se on kuollut tai se ei enää ole kantamalla ja vasta sitten ottavat uuden kohteen.

Käyttöohje

Peli käynnistetään avaamalla main.py tiedosto python3 tulkilla. Tämän jälkeen terminaalissa kysytään kenttä tiedoston nimi. Kenttätiedoston on oltava samassa kansiossa muiden tiedostojen kanssa. Tämän jälkeen peli käynnistyy. Pelin alkaessa pelaaja asettaa haluamansa tornit reitin varrelle, ei kuitenkaan reitin päälle. Rahatilanteensa pelaaja voi tarkastaa painamalla i-näppäintä jolloin aukeaa info. Tornit on numeroitu ja painamalla e-näppäintä, aukeaa tieto millaisia torneja on käytettävissä. Kun pelaaja on valmis tornien asettelun kanssa, uusi vihollisaalto kutsutaan n-näppäimellä. Kun viholliset on tuhottu tai ne ovat päässet maaliin, pelaaja voi taas asettaa torneja. Pelissä on myös niin sanottu salaominaisuus joka käynnistyy tietyssä tilanteessa s-näppäimellä.

Ohjelman rakenne

Ohjelma on jaoteltu seuraavasti. Enemy.py tiedosto sisältää luokan Enemy joka on kaikille vihollisille perusluokka, jossa määritellään yksittäisen vihollisen toimintaan liittyviä metodeja sekä alustetaan vihollinen. Enemy luokkaan ei liity itsessään muita luokkia eli se perii ainoastaan pelissä käytetyn pygame kirjaston perustoiminnot joiden avulla on mahdollista piirtää vihollinen ruudulle. Tämän lisäksi se perii tiedoston const.py johon on koottu perustietoja pelistä kuten ennalta määritettyjä värejä sekä FPS ja ruudun koko. Keskeisimpiä metodeja luokassa vihollisen alustamisen lisäksi on didEnemySurvive metodi jolla tarkastellaan pääsikö vihollinen maaliin ja metodi move jolla liikutetaan vihollista haluttuun suuntaan halutulla nopeudella sekä metodi killEnemy joka tuhoaa vihollisen. Näiden lisäksi löytyy muutama ns. get metodi joilla saadaan esille vihollisen perustietoja sekä ns. set metodi jolla taas muutetaan vihollisen tietoja.

Vihollisiin liittyy myös tiedosto enemies.py jossa on luokka Enemies joka käsittelee vihollisia suurempina joukkoina. Enemies.py perii luonnollisesti Enemy luokan jotta myös yksittäisen vihollisen metodit ovat käytettävissä. Enemies luokka on tarpeellinen kun käsitellään vihollisia

aalloissa, joina ne esiintyvät pelissä. Keskeisimpiä metodeja ovat moveEnemies joka käsittelee käytännössä sen että viholliset liikkuvat oikein kentällä, eli kääntyvät oikeissa kohdissa oikeaan suuntaan ja sen pääsikö vihollinen maaliin. Lisäksi löytyy metodi spawn joka luo viholliset. DrawEnemies metodi piirtää viholliset ruudulle. Myös Enemies luokasta löytyy get ja set metodeja.

Seuraavaksi tower.py luokka jossa luokka Tower käsittelee nimensä mukaisesti lähes kaiken joka liittyy torneihin. Jos torni tyyppejä olisi enemmän kuin kaksi tai ne eroaisivat toisistaan todella paljon olisi varmastikin hyödyllistä luoda luokka Towers joka käsittelisi torneja isompina joukkoina. Tower luokassa on myös tarpeellista käyttää Enemy luokan metodeja. Keskeisimpiä metodeja ovat towerSelected joka tarkastelee onko hiiri tornin päällä, tämä on tarpeellista tornien oston yhteydessä. Shoot metodi nimensä mukaisesti hoitaa tornilla ampumisen. MoveTower metodi liikuttaa tornia sen oston yhteydessä oikeaan paikkaan. Ampumisen kannalta tärkeät metodit distanceToEnemy laskee etäisyyden johonkin viholliseen, closestEnemy tarkastelee mikä vihollinen on tornia lähinnä ja enemyInRange taas sen onko jokin vihollinen tarpeeksi lähellä ammuttavaksi. Näiden lisäksi löytyy metodi joka tarkastelee onko torni vihollisten kulkureitin päällä. Myös Tower luokasta löytyy get ja set metodeita.

Luokka Player joka käsittelee pelaajan kannalta keskeisiä asioita kuten kuinka paljon rahaa on jäljellä tai mitä tapahtuu jos vihollinen pääsee maaliin tai jos niitä pääsee maaliin yli sallitun rajan. Player luokka pitää myös kirjaa siitä mitä kaikkia torneja pelaaja on asettanut reitin varrelle. Luokasta löytyy metodit joilla torneja voidaan lisätä pelaajalle ja liikuttaa niitä. GameLost metodi käsittelee pelin lopettamisen jos peli hävitään ja salaominaisuutta ei aktivoitu. GameWon metodi käsittelee pelin voittamisen. Tähän liittyen löytyy myös metodi enemyPassed joka käsittelee sen jos vihollinen pääsee maaliin asti ja mitä silloin tapahtuu. Metodi secret nimensä omaisesti aktivoi salaominaisuuden. Lisäksi on metodit initInfo ja drawInfo jotka alustavat ja piirtävät ruudulle pelaajan tiedot. Myös Player luokasta löytyy get ja set metodeita.

Tiedosto gameinitializer.py sisältää kaksi luokkaa Block ja Reader. Block on vain geneerinen palikka jota käytetään kentän alustamisessa. Tämän takia olisi ollut täysin turhaa tehdä sille omaa tiedostoa. Reader luokka vastaa kaikkien tietojen lukemisesta tiedostosta sekä pelikentän staattisista eli liikkumattomista osista. Keskeisimmät metodit ovat readFile joka hoitaa tiedostojen lukemisen. Metodit setTowerImages ja drawTowerNumbers asettavat tornien kuvakkeet ja niihin numerot eli luovat ns. torni katalogin josta tornit ostetaan. Metodit initExplanations ja drawExplanations hoitavat tornien selitysten alustamisen ja piirtämisen ruudulle. Metodi initializeGamefield alustaa vihollisten reitin. Myös Reader luokasta löytyy get metodeita.

Viimeisenä muttei vähäisempänä on main.py tiedosto jossa pyörii pelin ydin looppi. Luonnollisesti se perii kaikki luokat ja sen tehtävänä on koota kaikki yhteen ja kutsua tarvittavia metodeja tarpeen mukaan. Näin ollen main.py:ssä ei ole erillistä luokkaa. Aluksi alustetaan kaikki tarpeellinen. Tiedostossa käsitellään suurin osa näppäimistön painalluksista sekä hiiren painalluksista. Tässä tiedostossa myös kutsuttaan kaikki piirto komennot sekä muut mahdolliset metodien kutstut kuten vihollisten syntyminen jne.

Koska peli on toteutettu käyttämällä Pygame kirjastoa lähinnä graafisen käyttöliittymän aikaansaamiseksi monet luokat kuten Enemy ja Tower perivät pygame.sprite.Sprite:n alustuksen

jotta piirtämisen käsittely on mahdollista. Jokaisella luokalla on myös get ja set funktioita tai jompiakumpia.

Algoritmit

Kun lasketaan etäisyyttä tornin ja vihollisen välillä verrataan vihollisen ja tornin paikkaa ruudulla. Ruudun 0,0 piste on vasemmassa yläkulmassa joten on tärkeää huomioida suhteessa lasketaanko (tornin_x – vihollisen_x) vai (vihollisen_x – tornin_x), jotta etäisyys ei ole negatiivinen. Toinen vaihtoehto olisi laskea x ja y etäisyys aina samalla tavalla ja ottaa niiden itseisarvot. Loppujen lopuksi etäisyys lasketaan tutulla kaavalla sqrt(x^2 + y^2) = etäisyys.

Pelin kentän alustamisessa käytetty algoritmi saa parametereikseen tiedon kulmista joissa on käännös ja sen mihin suuntaan käännös tapahtuu, sekä aloituspisteen. Näiden tietojen perusteella algoritmi laskee paikan sekä sijoitettavan palan koon. Reitin leveys on vakio 50 pikseliä joka helpottaa hieman. Käytännössä vertaillaan aina seuraavan käännöksen x koordinaattia tämän hetkisen kulman x koordinaattiin ja y:lle vastaavasti sama. Esimerkiksi (tämä_x – seuraava_x)= leveys jos suunta on vasemmalle. Jos taas mentäisiin oikealle olisi leveys = (seuraava_x – tämä_x), sillä x koordinaatti kasvaa aina kun mennään ruutua oikealle ja vastaavasti alas niin y koordinaatti kasvaa. Jos kyseessä on viimeinen käännös verrataan vastaavasti x:ää ja y:tä ruudun leveyteen ja korkeuteen.Palikan leveyksiin ja korkeuksiin lisätään myös 38 pikseliä jotta myös käännösten kulmat peittyvät. X ja Y koordinaatit on myös määriteltävä palasille. Jos kuljetaan kasvavan x:n tai y:n suuntaan, vastaavasti koordinaatti on sama kuin annetulla kulmalla. Toinen kulmista on taas sama kuin annetulla kulmalla josta vähennetään 12 pikseliä johtuen siitä, että vihollinen kulkisi keskellä reittiä eikä reunaa pitkin. Tästä johtuu myös se että leveyteen ja pituuteen lisätään 38 pikseliä ja loput reitin leveydestä vähennetään sijainnissa. Jos mennään vähenevän x:n tai y:n suuntaan vastaava kulma on sijanti – leveys/korkeus +26. Toinen kulmista on taas annettu kulma josta vähennetään 12.

Kun tarkastellaan voiko pelaaja sijoittaa tornia haluamaansa paikkaan eli että se ei ole vihollisten reitillä joudutaan ottamaan huomioon reitin leveys sekä tornin leveys. Käytännössä laskenta toimii niin että verrataan keskenään reittiä joka koostuu paloista ja tornia, kulmien pareina. Esimerkiksi seuraavalla tavalla,merkitään tornia t:llä ja kenttää k:lla. T.ylävasen < K.ylävasen JA T.yläoikea < K.yläoikea. Jos tämä on tosi on siis mahdotonta että torni olisi reitillä. Tällä tavalla käydään kaikki tarvittavat kulmaparit läpi, jos jokin niistä on tosi torni ei ole reitillä ja se voidaan asettaa ehdotettuun paikkaan. Pygamessa on paljon törmäyksen havaitsemiseen käytettäviä valmiita metodeja mutta, koska projektissa pääpaino oli itse toteuttamisessa en niitä käyttänyt.

Pelin ydin loopissa käytetään jakojäännöstä hyväksi jotta saadaan halutut tapahtumat tehtyä tietyn ajan välein. Kun laskurin, johon lisätään jokaisella looppauksella FPS:n verran, jaetaan tietyllä luvulla saadaan asiat tapahtumaan vaikka sekunnin välein.

Tässä on esitetty algoritmit jotka ovat vaikeaselkoisia ilman taustaa. Lisäksi löytyy yksinkertaisia algoritmejä esimerkiksi kun lasketaan mikä vihollisista on lähimpänä tornia jolloin yksinkertaisesti tarkastellaan millä vihollisella etäisyys torniin on lyhin.

Tietorakenteet

Eniten käytetyt tietorakenteet ovat listat sekä pygamen group. Listan valitsin siitä syystä että sen kokoa on helppo muuttaa tarvittaessa esimkerkiksi lisättäessä torneja tai vihollisia. Pygamen group toimii käytänössä listan tavoin tosin hieman kankeammin joissain tilanteissa, se ei esimerkiksi tue samalla tavalla indeksointia kuin pythonin omat listat. Käytännön etuna groupissa on se että sillä on mahdollista piirtää yksinkertaisemmin kun vihollisia on monta. Pygamen groupiin tallennetaan pygame.Sprite objekteja. Sprite objekti taas sisältää ruudulle piirtämiseen ja muuten siihen liittyviä ominaisuuksia.

Tiedostot

Peli tarvitsee toimiakseen yhden tekstitiedoston jossa on tieto pelin perustiedoista, kuten kuinka paljon rahaa on alussa ja kuinka monta vihollista voi päästä läpi ennen kuin peli hävitään. Tiedosto sisältää myös tiedon vihollisista ja tornityypeistä joita pelissä on sekä pelikentästä eli vihollisten reitistä. Kyseinen tiedosto rakentuu seuraavasti.

Field:aloituskulma_x,aloituskulma_y,aloitussuunta

Enemy:vihollisten_määrä_aallossa,vihollisen_HP,vihollisen_nopeus

Corner:kulma_x,kulma_y,käännöksen_suunta

Tower:vahinko,nopeus,etäisyys,hinta,fiksu/tyhmä(1/0)

Player:rahat_alussa,vihollisraja_ennen_häviötä

Toisin sanoen tiedosto voisi näyttää vaikka seuravalta:

Field:0,0,down

Enemy:1,10,5

Enemy:2,10,4

Enemy:3,20,4

Enemy:4,16,8

Corner:0,200,right

Corner:300,200,down

Corner:300,400,left

Corner:100,400,down

Corner:100,500,right

Tower:4,10,300,100,0

Tower:3,20,200,200,0

Player:600,5

Tästä nähdään että kenttä tornit ja viholliset ovat hyvin kontrolloitavissa ja niiden tekeminen ei ole vaikeaa. Toisaalta on oltava tarkkana jotta tiedosto on oikenlainen, jotta se luetaan oikein ja se sisältää riittävästi tietoa. Sillä ei ole merkitystä missä järjestyksessä rivit ovat. Kuitenkin jokainen viidestä kohdasta on löydyttävä jotta peli toimisi oikein ja field ja player rivejä voi olla vain yhdet. Rivi jaetaan paloihin niin että tarkastellaan rivin ensimmäistä sanaa ja sitä vastaako se jotain tyyppiä. Esimerkiksi jos rivi alkaa /Corner sitä ei lueta kulmaksi. Tyypin jälkeen tulee ':' jonka jälkeen tarvittavat parametrit jotka erotetaan toisistaan pilkulla. Kaikki paitsi aloituspisteen ja kulman suunta parametrit luetaan int muotoisiksi. Suunnat ovat string muotoa.

Jokaisella rivillä täytyy olla vaadittu minimimäärä parametrejä jotta peli toimii. Toisaalta jos rivillä on ylimääräisiä pilkulla erotettuja parametrejä se ei haittaa. Eli käytännössä jos olisi uudemman version pelikenttätiedosto, jossa vaikkapa torneilla olisi enemmän parametrejä. Kyseistä kenttää voidaan silti käyttää mutta ilman lisäominaisuuksia.

Reitin eli kulmien suunnittelussa on tärkeää huomioida että toinen parametreistä on pysyttävänä samana jotta viholliset tunnistavat kulmat. Kulmilla on pienet marginaalit, jotka ovat sitä varten

ettei tietyllä nopeudella kulkevat viholliset mene kulmista ns.ohi. Kenttätiedosto on hyvin yksinkertainen, mutta toisaalta hyvin virheherkkä.

Testaus ja viat

Peliä oli tarkoitus testata pelaamalla ja tahallisesti tehdä asioita joiden ei pitäisi olla mahdollista ja katsoa toimiiko ohjelma toimitulla tavalla. Kun tornia yrittää asettaa reitille se ei onnistu vaan on yritettävä sijoittaa tornia uudestaan, tästä tulostetaan myös teksti terminaaliin jossa kerrotaan että kyseinen toiminto ei onnistu. Myöskään tornien osto ei onnistu vihollisaaltojen aikana. Kun halutaan nähdä pelaajan infot (i-näppäimellä) jotta tieto päivittyy, on se suljettava painamalla uudestaan i:tä ja taas avattava. Kentän tekemistä on testattu muutamilla eri konfiguraatioilla. Omissa testeissä en huomannut että kulmiin jäisi ns. tyhjää tilaa tämä kuitenkin saattaa olla mahdollista joissain kulmien yhdistelmissä.

Parannettavaa pelissä toki ja kehittää sitä voisi loputtomiin. Tällä hetkellä jos FPS:ää vaihtaa ampuma nopeus ja muut nopeudet muuttuvat myös.Iideaali tilanteessa FPS:n muuttaminen ei vaikuttaisi näihin. Tämän voisi korjata muuttamalla kaikki arvot joihin FPS liittyy, niin että ne riippuvat FPS ja muuttuvat suhteessa sen mukana. Virheherkkyytä voisi parantaa kenttäsuunnittelussa ja virhetilojen käsittelyä voisi parantaa. Tunnettu vika on jos vihollisen nopeus on liian suuri ja se ei osu ”mukavasti” kulman kohdalle eli +-5 pikseliä suuntaansa. Vihollinen menee kulmasta läpi ja se tulkitaan niin että se on päässyt maaliin. Toleranssia ei voi kasvattaa loputtomiin sillä muuten viholliset kääntyvät liian aikaisin jolloin ne eivät enää kulje piirrettyn reitin päällä. Myös reittiä voisi leventää jolloin myös toleranssi voisi olla suurempi, mutta mielestäni se ei poista ongelmaa vaan on tilapäisratkaisu. Tällä hetkellä pelattavuus ei ole paras mahdollinen sillä vihollisten, tornien ja alkurahojen balanssia ei käytänössä ole kovinkaan paljoa mietitty. Tämän voisi korjata vain kokeilemalla milloin kaikki on hyvässä balanssissa. Info tekstien ja tornikatalogin paikka on aina sama joten ne mahdollisesti ovat reitin tiellä. Tämän voisi korjata niin että tekstit ja katologi ovat tietyssä pisteessä mutta jos siinä kohtaa on kenttää ne liikkuvat johonkin suuntaan niin kauan kunnes ne eivät ole enää tiellä. Ongelma ei ole niinkään suuri tekstien kanssa koska ne saa piiloon.

Parhaina kohtina pidän pelin luettavuutta tiedostosta. On myös mukavaa että halutessaan vaikeustasoa on helppo muokata. Myös se että lopulta sain tehtyä kaksi erilaista tornityyppiä on mukavaa. Heikkouksina pidän sitä että pelissä ei ole minkäänlaista valikkoa. Kenttien virheherkkyys on myös ikävää, toisaalta kun sen tiedostaa kenttien suunnitelu ei ole vaikeaa. Jos jatkaisin projektia tornit ampuisivat myös liikkuvia ammuksia.

Suunnitelman toteutus ja aikataulu

Suunnitelmat toteutuivat mielstäni pääpiirteittäin hyvin. Projektin tehtävän annossa sanottiin että kannattaa painottaa asetustiedostoista lukuun joka mielestäni totetutui hyvin. Graafista ilmettä oli alunperin tarkoitus tehdä kauniimmaksi. Ideana oli että jos aikaa jää niin silloin graafinen puoli toteutetaan nätimmin mutta aika ei riittänyt. Alunperin suunnitelmissa oli käyttää Box2D ja SFML

kirjastoja joita olen käyttänyt jo aiemmin, mutta loppujen lopuksi päädyin käyttämään pygamea sillä tarvitsin kirjaston vain piirtämistä varten.

Suurin osa luokista luotu ja yksinkertaiset metodit ja alustukset. Ohjeet pygamen asentamiseen readme:ssä(31.3)

Kentän ja tiedoston luvun kehittäminen (5.4)

Vihollisten syntyminen ja vihollistaaltojen toiminta (14.4)

Tornit ja niiden perusominaisuudet ja toiminnallisuudet (19.4)

Tornien liikutus luotu ja toimintaa parannettu (3.5)

Player luokkaan luotu lisää metodeja. Tornien liikutus toimii niin kuin pitää ampuu silloin kuin pitää. (4.5)

Tornien luominen toimii oikein eli torni katalogin kuvakkeet eivät liiku vaan luodaan uusi torni(5.5)

Myös pelaajan tiedot luetaan tiedostosta. Kentän luku korjattu. Kenttiä paranneltu. (6.6)

Aikataulullisesti projektin olisi voinut aloittaa aikaisemmin mutta omien kiireiden takia aikaa oli hyvin rajallisesti. Toisaalta itselleni tämä ei tullut yllätyksenä.

Yhteenveto

Loppujen lopuksi olen tyytyväinen peliin. Peli toimii perusoiminaisuudet jotka oli tarkoitus toteuttaa toimivat. Parannettavaa tietysti löytyy aina ja kehittää peliä voisi loputtomiin. Laajennusten ja lisäosien suhteen uskon että peli mukautuisi hyvin varsinkin tiedostosta luettavuuden ansiosta myös esimerkiksi uudemmat kentät, oikein toteutettuina, toimisivat vanhojen versioiden kanssa. Graafista ilmettä voisi parantaa sekä tornit voisivat ampua ammuksia. Jos jatkaisin projektia seuraava askel olisi tehdä liikkuvat ammukset. Mahdollisesti main.py:tä olisi voinut siistiä ja osia siitä olisi voinut yhdistää muihin luokkiin jolloin se olisi helpompi lukuisempaa. Koodia muutenkin olisi voinut siistiä ja yhtenäistää enemmän.

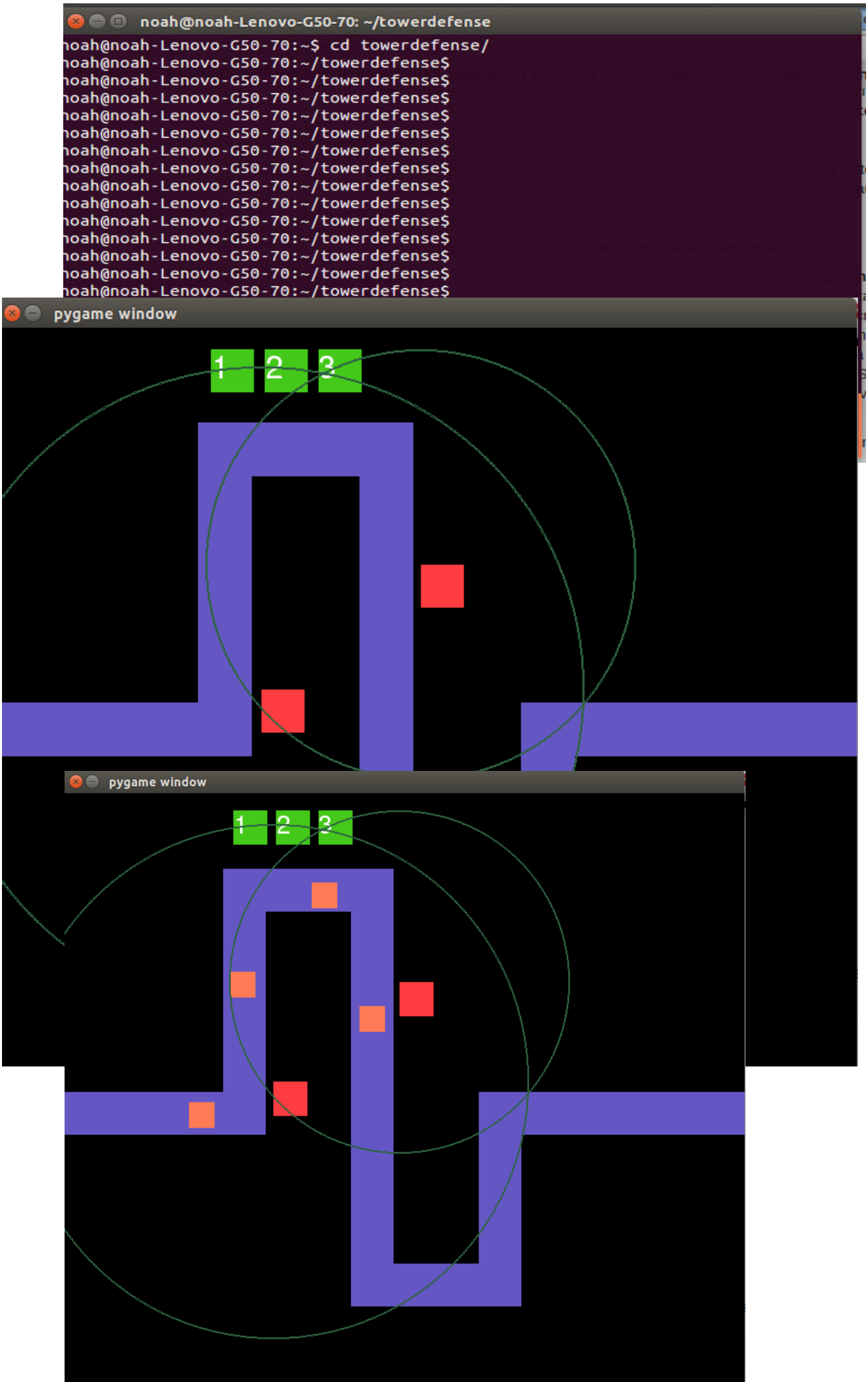
Viitteet

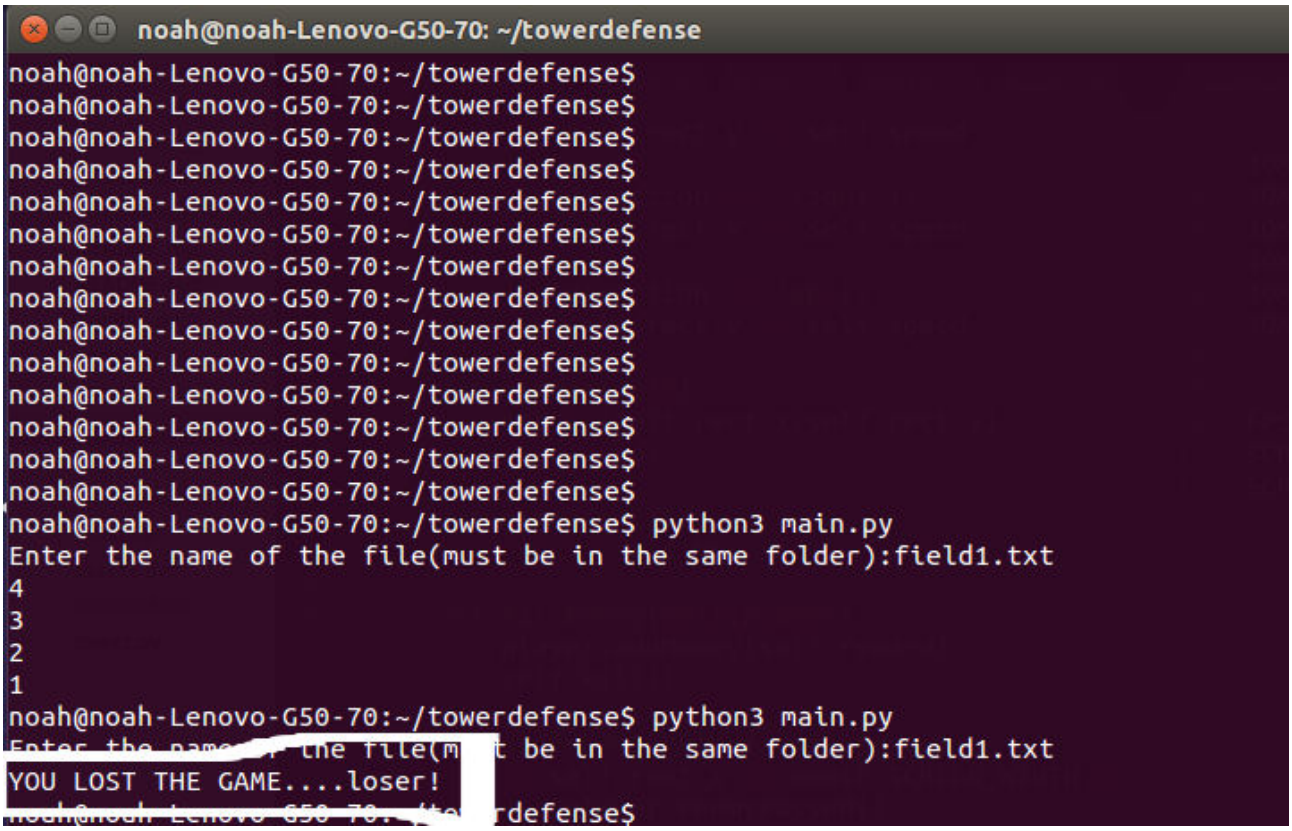
http://www.pygame.org/docs/

https://docs.python.org/2/tutorial/

Making Games with Python & Pygame: Al Sweigart 2012

Liitteet


noah@noah-Lenovo-G50-70: ~/towerdefense
noah@noah-Lenovo-G50-70:~$ cd towerdefense/
noah@noah-Lenovo-G50-70:~/towerdefense$
pygame window
1
2
3
pygame window
1
2
3


noah@noah-Lenovo-G50-70: ~/towerdefense
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$
noah@noah-Lenovo-G50-70:~/towerdefense$ python3 main.py
Enter the name of the file(must be in the same folder):field1.txt
4
3
2
1
noah@noah-Lenovo-G50-70:~/towerdefense$ python3 main.py
YOU LOST THE GAME....loser!