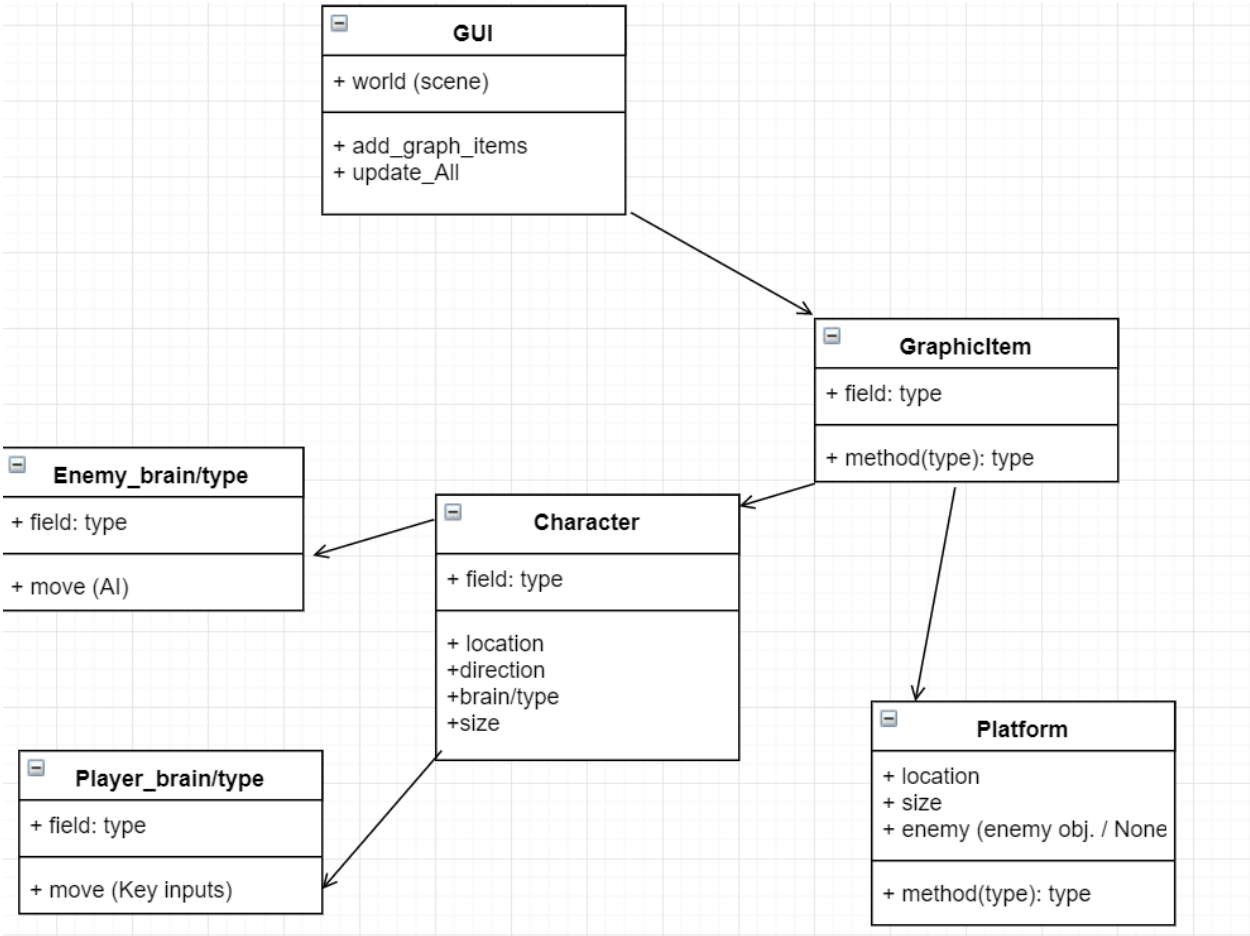
1. Program’s structure plan



Alustava suunnitelma rakenteesta. GUI, eli graafinen käyttöliittymä, eli se mitä pelaaja näkee luo maailman (scene:n) mihin grafiikkaobjektit asetetaan. GraphicItem luokka luo graafisen objektin jokaiselle objektille, joka lisätään maailmaan. Alustavasti meillä on character sekä platform objekteja joista character objektilla on joko vihollisen tai pelaajan aivot tai tyyppi joka taas määrää sen, liikkuuko hahmo key-inputista vai AI:n perusteella ja miltä hahmo näyttää jne. Platform objektit luodaan omasta luokastaan, ja niillä voi olla vihollinen tai ei. Jos sillä on vihollinen niin vihollinen ’spawnaa’ samanaikaisesti platform objektin kanssa sen päälle. Rakenne on aika yksinkertainen ja kaukana täydellisestä, mutta se tulee todennäköisesti tekovaiheessa muuttumaan.

2. Use case description

Käyttäjä käynnistää projektin ‘main’ tiedoston joka luo main-ikkunan johon asetetaan oletuksena start-scene, eli ts. alkunäkymän start ja exit napeilla. Käytäjä painaa start-nappia jolloin siihen yhdistetty metodi asettaa ikkunaan uuden scenen joka sisältää nyt tason objekteineen jollain alkuarvoilla. Eli käyttäjä on aloittanut pelin. Pelin taustaa lukuunottamatta kaikki muut ruudulla näkyvät asiat ovat grafiikka-objekteja joilla on ainakin sijainti. Kun pelaaja liikuttaa hahmoa, keyPressEventteihin sidotut metodit vaihtavat hahmon sijaintia. Oikealle liikuttaessa tarpeeksi, vaihdetaan myös kaikkien ruudulla näkyvien objektien sijaintia vasemmalle hahmoon nähden joka luo illuusion liikkumisesta maailmassa. Samalla renderöidään lisää objekteja jottei ruutu tyhjenisi kokonaan. Pelaaja pomppii tasoilta toiselle välttäen tai tuhoten vihollisia kunnes tarpeeksi oikealle liikuttuaan renderöinään maalia kuvaava grafiikka-objekti jota koskettamalla pelin voittaa.

3. Algoritmit

Pelissä tulee olla tasoja tarpeeksi ja tietty tarpeeksi lähellä, toisiaan jotta niitä pitkin voi kulkea. Muutenhan peli olisi mahdotonta läpäistä. Myös vihollisten täytyy olla tasojen päällä tai ne putoavat kuolemaan. Viholliset syntyvät aina tason päälle. Tiedämme (ohjelma tietää) tason sijainnin, sen leveyden sekä korkeuden. Vertaamalla vihollisen koordinaatteja tason päiden koordinaatteihin varmistamme, ettei vihollinen ylitä näitä ja tipu alas. Uusien tasojen syntyminen on tietenkin jossain määrin satunnaista mutta siinäkin ehtona on, että hahmo voi hypätä sen hetkiseltä tasolta lähimmälle seuraavalle. Eli laskemme ajan jonka hahmo on ilmassa hypättyään tasolta. Tämä ilmassaoloaika riippuu tietenkin myös siitä, kuinka pitkään hän tippuu eli kuinka paljon matalammalla tai korkeammalla seuraava taso on verrattuna edelliseen. Sen ajan mitä pelaaja on ilmassa, on hänellä aikaa liikkua x-tasossa, eli seuraavan tason maksimi x-etäisyys edellisestä tasosta riippuu suoraan sen korkeudesta edelliseen nähden. Sanotaan että hypättyään pelaajalla menee t-sekuntia korkeimpaan kohtaan mistä se alkaa pudota kiihtyvyydellä ’g’. Tällöin matkan y-putoamiseen menee aika $\sqrt{\frac{2*y}{g}}$

Eli pelaaja ehtii liikkua x-akselilla tämän ajan + t. Tämä kaava antaa meille rajoitteen seuraavan tason korkeudelle ja x-etäisyydelle. Pelaajan nopeus x-akselilla on jokin vakio ja hyppyfysiikka toteutetaan kaavalla v = v_0 – g*t missä v on nopeus v_0, alkunopeus, g = gravitaatio ja t = aika. Eli pelaaja nousee ylös alkunopeudella v_0 joka pienenee ajan kuluessa muuttuen negatiiviseksi, jolloin pelaaja putoaa, kunnes se osuu laavaan tai tasolle ja v laitetaan nollaan.

4. Data structures

Pelissä esiintyvät objektit, (hahmo, viholliset ja tasot) tallennetaan kullekin objekti-tyypille luotuun listaan johon lisätään ja josta poistetaan objekteja sitä mukaa kun tarve

on. Objektit itsessään sisältävät tarvittavat kentät ja suhteet toisiinsa. Kaikilla objekteilla on joku sijainti eli koordinaatit ja koko. Pelaaja ja vihollisobjektit ovat molemmat ’character’ objekteja joilla ’health’, sijainti, suunta jne. sekä näiden lisäksi esim. aivot tai tyyppi. ’Aivojen’ tai ’tyyppien’ perusteella vihollinen ja pelaaja-hahmo erotetaan toisistaan missä vihollisaivot vastaavat koodattua AI:tä ja pelaaja-aivot vastaavat reagoimista nuolinäppäimiin ja välilyöntiin.

5. Libraries

Projektissa käytetään PyQt-kirjastoa tai ’frameworkia’. Sen avulla luomme koko pelin kannalta enemmän kuin oleellisen graafisen käyttöliittymän. Sen tarjoama QKeyEvent-luokka mahdollistaa myös kätevän käyttäjän sisääntulon käsittelemisen koodissa. PyQt tarjoaa myös QTimer luokan jonka avulla voidaan toteuttaa pelin-looppi.

6. Schedule

Alustava suunnitelma pelin toteutuksesta ja aika-arviot:

1. Aloitusnäytön toteutus, eli pelin käynnistys avaa ikkunan kahdella napilla ja molempien nappien painallus rekisteröidään. 1-2h max?
2. Nappien toiminnallisuuden toteutus. ’Exit’-nappi lopettaa ohjelman ja ’Start’-nappi Aloittaa peli-loopin. 1-2h?
3. Pelaajan-hahmon toiminnallisuus. Luodaan grafiikkaobjekti, näppäinten painamiset muuttavat hahmon sijaintia haluamallamme tavalla. 2-8h?
4. Taso-objektien toteutus. Pelaaja voi liikkua tasojen päällä ja hyppiä niiltä toisille. 3-8h?
5. Vihollisten toteutus. (Liikkuvat edestakaisin tasolla kävelemättä yli) Pelaaja voi aiheuttaa niille vahinkoa ja toisinpäin.2-5h?
6. Itse tason taustan toteutus. Maa laavaa joka aiheuttaa kuoleman, ja tausta muuten halutun näköinen. Elämien määrä näkyy, ja ’health’ tilanne. 3-6h?
7. Scroll-efekti, eli uusien objektien luominen ja vanhojen siirtäminen vasemmalle luoden illuusion liikkumisesta maailmassa. 3-7h?
8. Maalin luominen ja pelin loppuminen se saavutettaessa. 1-4h?

7. Unittesting plan

Haluamme testata kokonaisuudessaan seuraavat asiat:

- Pelin voi aloittaa painamalla start-nappia ja sulkea painamalla exit-nappia.

- Pelaaja voi liikkua, ja aiheuttaa sekä ottaa vahinkoa oikea määrä oikeista asioista (viholliset, laava)
- Pelaaja voi liikkua tasoilla ja ylipäätään saavuttaa maalin.
- Peli loppuu kuolemasta, sekä voitosta

Näiden testaaminen lienee kannattavinta toteuttaa yhdessä unittestien ja PyQt:n tarjoaman QTest-luokan avulla. QTest-luokka tarjoaa listan eri metodeja joilla voimme itse testissä klikata ja painaa nappeja ja verrata näiden antamia lopputuloksia haluttuihin.