1. class Gui

Piirtää varsinaisen peli-ikkunan sekä aloitusikkunan.

perii QMainWindow (PyQt5.QtWidgets.QMainWindow)

__init__(self):

- alustaa pääikkunan koon
- määrittelee yhteyden GameField ja DrawGame olioon
- alustaa mainwindowin (start screen)
  - pohjana kuva pelistä
  - keskellä QPushButton('Start Game')
    - painaminen yhdistää start_gameen
    - toimii hiirellä

start_game(self):

- valmistelee varsinaisen pelin piirtämisen aloittamisen
  - laittaa napin (start game) pois päältä ja pois piirrosta
  - muuttaa piirtoalueen kokoa tarvittaessa
  - aktivoi pelin piirtämisen, muokkaa GameField olion pelin käynnissä oloon liittyvää arvoa

update_game(self):

- varsinainen pelin piirtäminen DrawGame luokan avulla

keyPressEvent(self,event):

- keyPressEventin määrittely (luokasta QtCore QEvent)
- signaalin emittaus determine_key metodille (pyqtSignal(QEvent))

determine_key(self,event):

- Tarkastaa, oliko painettu key space, right arrow (D) tai left arrow (A)
- Jos oli, niin muuttaa GameField olion vastaavien nappuloiden painokertoja ilmaisevia arvoja

end_game(self):

- Kun peli päättynyt piirtää pelin lopputuloksen näytölle
  - peli päättyminen GameField olion arvona
- Alustaa pelin alkutilanteeseen, kun käyttäjä painaa nappia Continue
  - emittoi signaalin reset_gui metodille (pyqtSignal())

reset_gui(self):

- Alustaa ohjelman lähtötilanteeseen

2. class GameField
    - luokkamuuttujat: Space(int), Right_arrow(int), Left_arrow(int), Game_running(Bool), Fail(Bool)

__init__(self):

- lista StaticObject, Enemy olioista (alussa tyhjä)
- player on kytkentä pelaajaolioon
- pelialueen leveys Width- muuttujana
- space, right_arrow, left_arrow muuttujat

parse_gamefield(self,file):

- asettaa gamefieldin leveyden
- asettaa pelaajan self.player
- asettaa listaan staattiset objektit ja viholliset filen perusteella
- luettavan filen esimerkkimuoto löytyy mm. yleissuunnitelmasta

add_space_key(self):

- tarkoitettu kutsuttavaksi guista käsin, lisää spacen painalluksen gamefieldiin

add_left_key(self):

- tarkoitettu kutsuttavaksi guista käsin, lisää left_arrow arvoa gamefieldiin

add_right_key(self):

- tarkoitettu kutsuttavaksi guista, lisää right_arrow gamefieldiin

update_objects_positions(self):

- päivittää kaikkien vihollisten ja pelihahmon paikkaa: kutsuu siis osametodeja
- kutsutaan DrawGamesta käsin

update_player_pos(self):

- päivittää pelihahmon paikkaa sivusuunnassa (move-metodilla) sekä hyppäyttää hahmon kutsuen jump-metodia
- kutsuu move-metodia, jos vain right_arrowin tai left_arrowin arvo muuttunut. Jos myös space muuttunut kutsuu jump-metodia. Hyödynnetään luokkamuuttujia
    - ennen kuin kutsuu movea, tarkastaa että jumping(Bool)= False ja falling(Bool) = false
      kts.class Player
    - jos jumping tosi, kutsuu jump metodia
    - jos falling tosi, kutsuu fall metodia

update_enemy_pos(self):

- päivittää vihollisen paikkaa sivusuunnassa move -metodilla
- vihollisen nopeus vaikuttaa
- vihollisen reverse(Bool) arvo vaikuttaa suuntaan kts. class Enemy
    - True = oikealle
    - False = vasemmalle

move(self,object,direction):

- liikuttaa objektia haluttuun suuntaan tietyllä nopeudella
    - objectin tyyppi ja sen x_speed muuttujan arvo määrittelee liikuttamisnopeuden (kuinka monta pikseliä kerralla)
- tarkastaa ensin, että sijainti on mahdollinen check_position metodin avulla
    - paluuarvon perusteella muuttaa objektin sijaintia position luokan change_position -metodin kautta
- jos objektin (pelaajan) leveli (player.level) on ei ole yhtä suurempi kuin alapuolella olevan objektin leveli, laitetaan falling Trueksi. Ideana se, että pelihahmo ei jää liikkumaan ilmaan
    - leveli vertailu: otetaan handle objectiin get_static_object_at_position ja kutsutaan sille get_object_level. Verrataan playerin level -kentän arvoon

jump(self,player,direction):

- hyppäyttää pelaajaa, jos mahdollista
    - jos jumping == False, previous_position = player.position ja jumping = True
    - suunta ylös
        - jos jump_y < max_height, niin yritetään hypätä jump_height verran ylös (kts. Player)
        - jos jump_y >= max_height, niin asetetaan falling Trueksi (aloitetaan putoaminen) ja player leveliä lisätään yhdellä
    - suunta oikealle
        - jos jump_y < max_height, yritetään hypätä jump_y verran ylös ja liikkua oikelle sivulle jump_x verran
            - jos check_position = False palataan lähtöpaikkaan eli hyödynnetään previous_position muuttujaa ja asetetaan jumping=False (tällä yritään estää jumiutumista, jos hypätään liian läheltä staattista objektia oikealle)
            - jos hyppy mahdollinen, liikutetaan pelaajaa oikealle jump_x ja ylös jump_y verran (siis alaspäin pikseleinä)
        - jos jump_y >= max_height, niin falling trueksi ja level 1 korkeammaksi
    - suunta vasemmalle
        - jos jump_y < max_height, yritetään hypätä jump_y verran ylös ja liikkua vas. sivulle jump_x verran
            - jos check_position = False palataan lähtöpaikkaan eli hyödynnetään previous_position muuttujaa ja asetetaan jumping=False (tällä yritään estää jumiutumista, jos hypätään liian läheltä staattista objektia vasemmalle)
            - jos hyppy mahdollinen, liikutetaan pelaajaa oikealle jump_x ja ylös jump_y verran (alaspäin pikseleinä)
        - jos jump_y >= max_height, niin falling trueksi ja level 1 korkeammaksi

Info: jumping muuttujan tarkoitus estää, että hahmoa ei voi ohjata sivulle hyppysekvenssin aikana

fall(self,player,direction):

- yritetään tiputtaa pelaajaa fall_y verran alas
    - jos kohdassa johon pelaaja yritetään tiputtaa, on vihollinen eli get_enemy_at_position paluuarvo ei ole None, niin vihollinen tuhotaan
        - vihollisen destroyed muuttujan arvoksi True
    - testataan uuden position mahdollisuus check_position metodilla
        - jos paluuarvo True, muutetaan pelaajan sijaintia fall_y verran alas
        - jos paluuarvo False, Falling = False
    - pudotuksella myös sivuttaissiirtymä fall_x joko oikealle tai vasemmalle

get_static_object_at_position(self,position_x):

- käydään staattiset objektit läpi ja verrataan position_x arvoa stattisen objektin x_min ja x_max arvoon (jos arvo on välillä, niin positio_x on kyseisen objektin päällä)
    - tulee huomioida, että objekteja useita päällekkäin ja palautetaan korkeimmalla oleva objekti (tarkoitus käyttää get_object_levelin kanssa)
    - jos objektia ei ole paikassa, palautetaan None

get_enemy_at_position(self,position_x):

- käydään viholliset läpi verraten positio_x arvoa vihollisen x_min ja x_max arvoihin
    - jos vihollinen löytyy, palautetaan vihollisobjektin
    - muulloin palautetaan None

get_object_level(self,object):

- palautetaan objektin leveli (kts. GameObject)
- jos objekti None, palautetaan 0

check_position(self,object,position):

- jos kohdassa on este tai pelialueen reuna, palauttaa arvon False
    - jos kyseessä vihollinen muuttaa reverse(Bool) arvon
- jos kyseessä pelaaja ja kohdassa vihollinen (ei tuhottu, eli destroyed False kts Enemy, palauttaa arvon False ja muuttaa Game_running = False ja Fail = True
- jos kyseessä vihollinen ja kohdassa pelaaja, sama kuin yllä

- jos kyseessä on pelaaja ja kohdassa on ns. näkymätön objekti (kuilu), palauttaa arvon Fail ja muuttaa Game_running= False ja Fail=True
- jos kyseessä on pelaaja ja kohdassa maaliobjekti(finish), palauttaa True ja muuttaa Game_running =False ja Fail=False

Huom. vaikuttaa tällä hetkellä aika raskaalta metodilta ja sijaitsee monen edeltävän metodikutsun takana

3. class DrawGame
   - perii pyqt5 piirtämiseen liittyviä luokkia

__init__(self):

- gamefield muuttujana GameField olio

draw_mainframe(self):

- piirtää gamefieldin stattiset objektit ja ”pelikentän alustan”
- hyödyntää vahvasti objektien positiota (kts. class position ja StaticObject)
- staticobjectit visible piirretään suorakulmiona (väri ruskea)
- alusta piirretään tummana ja sitä ei piirretä kohtiin joissa on invisible staticobject
- finish piirretään vihreänä suorakulmiona muuten samanlainen kuin visible

draw_other_objects(self):

- piirtää gamefieldin pelihahmon ja viholliset
  - hyödyntää vahvasti objektien positiota (DynamicObject ja class position)
- päivittää ensin kaikkien liikkuvien olioiden paikat kutsumalla gamefield update_objects_positions

4. class GameObject

__init__(self):

position (alussa None)

type (määritellään uudelleen alaluokissa)

level

set_object_level(self,level):

- muutetaan level arvo vastaamaan pelikentän sisältävässä filessä olevaa

set_position_object(self,position):

- asetetaan gameobject-oliolle positio-olio (kts.positio)

class StaticObject

Width (staattisten objektien leveys)

Height (staattisten objektien korkeus)

- class Visible
    - type : visible_object
- class Invisible
    - type : invisible_object
- class Finish
    - type: finish_object

class DynamicObject

speed _x (määritellään uudelleen alaluokissa)

set_speed(self,value):

- alustaa nopeuden, jolla liikutaan x-suunnassa

class Player

- type: player
- speed_x suuri luku (ilmaisee move nopuden)
- jump_x (ilmaisee x-siirtymän hypätessä)
- fall_x (ilmaisee x-siirtymän pudotessa)
- jym_y (ilmaisee y-siirtymän hypätessä
- falling (alussa False)
- jumping (alussa False)

class Enemy

- type: enemy
- destroyed (Bool)

- class Slow
  - speed_x pienempi luku

- class Fast
  - speed_x suurempi luku

5. class Position

   Luokkamuuttujat:

   Distance_x (etäisyys piirtoalueen reunasta pikseleinä, ns. 0 paikan objektille = kaikista vasemmalla olevalle objektille)

   Distance_y(etäisyys piirtoalueen yläreunasta, ilmastaan nollatason objektien etäisyytenä)

- sisältää objektin sijainnin reunapisteet (pikselit)
- jokaisella Gameobjektilla position olio

__init__(self):

- x_min vasemman rajan koordinaatti
- x_max oik. rajan koordinaatti
- y_min alareunan koord.
- y_max yläreunan koord.

set_position(self,x,y,level,object_type):

- kutsutaan kun parsitaan filestä pelikenttää ja luodaan objektit
  - x ja y ovat filessä ilmaistut sijainnit
  - level on objektin filessä ilmastu taso (esim. 0, 1,2)
    - kertoo, kuinka monta staattista objektia on tämän objektin alla
  - paikka y_min saadaan kertomalla leveli staattisten objektien korkeudella ja vähentämällä tämä arvo Distance_y:stä
  - y_max saadaa Y_min – objektin tyypille ominainen korkeus
  - x_min saadaan kertomalla x stattisen objektin leveydellä ja lisäämällä Distance_x
  - x_max saadaan lisäämällä x_min arvoon kyseisen objektin leveys

Huom. tällainen algoritmi toimii vain, jos staattisten objektien leveys on vakio. Pelaajahahmon ja vihollisten leveyden tulee käytännössä olla pienempiä kuin staattisten objektien leveyden. Jotta tällöin invisible objektien yli pystyttäisiin hyppäämään, tulee olla kenttäsuunnittelussa visible objekteja kaksi rinnakkain ja hypyn pitää sivusuunnassa olla suurempi kuin 1 staattisen objektin leveys. Staattisten objektien korkeuden pitää tietenkin olla selvästi matalampi kuin hypyn