Haaga-Helia
University of Applied Sciences

Ohjelmistokehityksen teknologioita - *S*eminaarityö

Pygame peli

7 Käyttöliittymät ja pelit

Hilda Väänänen

14.4.2022

Sisältö

Tiivistelmä

Tarkoitus oli luoda peli, jossa pelaaja kerää pisteitä ja väistelee vihollisia. Pelaaja saa pisteitä keräämällä esineitä, joita ympäristöstä löytyy. Pelaaja voi puolestaan menettää elämän osumalla ansaan tai viholliseen. Peli päättyy, kun pelaaja menettää kaikki elämänsä. Peli toteutettiin Pygame-työkalun avulla ja Python-ohjelmointikielellä. Apuna oppimisessa käytin netistä löytyviä videoita, sekä Pygamen-dokumentaatiota.

1 Johdanto

Työni tavoitteena oli tutustua 2D pelin tekemiseen Pygame-työkalun avulla, tutustumalla sen tarjoamaan kirjastoon, sekä hyödyntää sitä pelin kehittämisessä. Apuna oppimisessa käytin netistä löytyviä videoita ja Pygamen dokumentaatiota.

Aloitin työskentelyn katsomalla läpi ja ottamalla mallia löytämästäni videosta ”Python Platfromer Game Tutorial for Beginners”, jossa käytiin läpi erilaisten pelin elementtien luomista. Video on julkaissut ”freeCodeCamp.org”. Videon ohjeistuksella sain luotua peliympäristön (tausta, maa), pelattavan hahmon, joka liikkuu ja hyppii näppäimistöltä tulevien komentojen mukaan, sekä ansan, johon törmättäessä pelattava hahmon reagoi. Mainittuja elementtejä tehtäessä käytiin läpi kuvien tuonti tiedostoista peliin, syötteiden saaminen näppäimistöltä, animointia, painovoiman luominen, sekä törmäyksien luominen.

Videon katsomisen jälkeen aloin tutustumaan tehtyyn koodiin tarkemmin hyödyntäen Pygamen dokumentaatiota sekä tekemiäni muistiinpanoja. Sovelsin oppimaani tekemällä lisäyksiä peliin, kuten lisäämällä kerättävät elementit, pisteiden keräämisen, osumien (damage) saamisen sekä NPC:n. Lisäksi kokeilin parannella pelattavan hahmon liikkumista ja muutin ympäristön ja hahmon ulkonäköä.

Pelin kuvat ja taide ovat peräsin seuraamani harjoituksen GitHub-repositorysta, josta latasin harjoitukseen liittyvät tiedostot.

2 Käytetyt tekniikat

Seuraavaksi käyn tarkemmin läpi työssä käytettyjä tekniikoita kertomalla tarkemmin joistakin koodin ja pelin kannalta keskeisimmistä funktioista ja luokista.

2.1 Pygame-työkalu

Pygame on peliohjelmointikirjasto, joka tarjoaa työkalut pelien kehittämiseen Pythonilla. Se sisältää useita eri objekteja ja moduuleja, joita voi hyödyntää pelin kehittämisessä. Käyn läpi kohtia koodista, joissa olen hyödyntänyt Pygamen kirjastoa.

2.1.1 Pelinäytön valmistelu

Pygamen ”display”- moduulia käytettään kontrolloimaan näyttöikkunaa. Käytin sen tarjoamia funktioita pelinäytön nimeämiseen ja koon määrittämiseen (https://www.pygame.org/docs/ref/display.html).

```
9    pygame.display.set_caption("Platformer")
```

Kuva 1. Pelinäytön nimeäminen.

```
15    window = pygame.display.set_mode((WIDTH, HEIGHT))
```

Kuva 2. Pelinäytön koko.

2.1.2 Kuvien hakeminen

Pygamen tarjoama ”image”- moduuli on tarkoitettu kuvien siirtämiseen tiedostoista ja käytin sen ”load”-ominaisuutta kuvien tuomiseen tiedostoista peliin (https://www.pygame.org/docs/ref/image.html). Koodissa halutun polun nimi talletetaan muuttujaan, jonka jälkeen annetusta polusta löytyvät tiedostot talletetaan listaan. Lista käydään läpi ja jokaisesta kuvasta luodaan ”sprite_sheet”, jotka jaetaan yhtä-suuriin osiin leveyden mukaan. Kuvien saamiseksi ruudulle luodaan ilmentymä ”Pygame.Surface”-objektista, eli luodaan pinta, jolle voi piirtää grafiikkaa (https://www.pygame.org/docs/ref/surface.html). Pinnalle määritellään yhtä ”sprite_sheet”- osan kokoa vastaava kehys käyttäen ”Pygame.Rect”- objektia, jolle annetaan parametrinä sen kohdan x-koordinaatti ja y-koordinaatti, josta kehys otetaan, sekä haluttu kehyksen koko (https://www.pygame.org/docs/ref/rect.html). Kehyksen sisältö kopioidaan aikaisemmin luodulle pinnalle ja lisätään ”Sprites”-listaan. Kuvien hakuun ja siirtämiseen liittyvä funktio on nähtävissä kuvassa neljä (kuva 4). Kuvien (sprite_sheet) jakamista osiin havainnollistetaan kuvassa kolme (kuva 3).

Kuva 3. ”Sprite_sheet” jakaminen osiin.

```python
def load_sprite_sheets(dir1, dir2, width, height, direction=False):
    path = join("assets", dir1, dir2)
    images = [f for f in listdir(path) if isfile(join(path, f))]

    all_sprites = {}

    for image in images:
        sprite_sheet = pygame.image.load(join(path, image)).convert_alpha()

        sprites = []
        for i in range(sprite_sheet.get_width() // width):
            surface = pygame.Surface((width, height), pygame.SRCALPHA, 32)
            rect = pygame.Rect(i * width, 0, width, height)
            surface.blit(sprite_sheet, (0,0), rect)
            sprites.append(pygame.transform.scale2x(surface))

        if direction:
            all_sprites[image.replace(".png", "") + "_right"] = sprites
            all_sprites[image.replace(".png", "") + "_left"] = flip(sprites)
        else:
            all_sprites[image.replace(".png", "")] = sprites

    return all_sprites
```

Kuva 4. Kuvien haku tiedostoista, jakaminen osiin ja siirtäminen taustoille.

Kuvien hakemisen jälkeen käytetään luotua funktiota ”flip” (kuva 5) kääntämään kuvat tarvittaessa toisin päin, jolloin esimerkiksi hahmon animaatioon liittyvistä kuvista (kuva 3) ei tarvitse luoda käänteistä versiota, jotta kävelyanimaatio saataisiin toimimaan kumpaankin suuntaan. ”Flip”- funktiota käytetään, jos ”load_sprite_sheets”-funktion ”direction” parametrille annetaan sitä kutsuttaessa arvo ”True”.

```python
def flip(sprites):
    return [pygame.transform.flip(sprite, True, False) for sprite in sprites]
```

Kuva 5. Funktio, joka käyttää ”pygame.transform.flip”- moduulia spriten kääntämiseen. Boolean arvot ”True” ja ”False” kontrolloivat sitä käännetäänkö sprite x-akselilla ja y-akselilla. (https://www.pygame.org/docs/ref/transform.html#pygame.transform.flip)

2.1.3 Törmäykset

Törmäys ominaisuuden luomiseen käytetään Pygamen ”sprite”-moduulin ”colllide_mask” ominaisuutta, jonka avulla tarkastetaan kahden spriten törmäys, tarkistamalla niiiden ”maskien” (mask) päällekkäisyyden. Ominaisuuden käyttämiseksi spritet tarvitsevat attribuutin ”rect”, mutta ”mask” on vapaaehtoinen (https://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_mask). ”Mask”- attribuutti kuitenkin mahdollistaa törmäysten tarkemman määrittelyn ja voi siten auttaa luomaan parempia törmäyksiä. ”Mask” luodaan annetusta pinnasta ”mask”- moduulin ”from_surface” ominaisuuden avulla (kuva 6).

```
self.mask = pygame.mask.from_surface(self.image)
```

Kuva 6. (https://www.pygame.org/docs/ref/mask.html#pygame.mask.from_surface)

2.1.4 Komennot näppäimistöltä

Hahmon saaminen liikkumaan näppäimistöltä tulevien komentojen mukaan onnistui hyödyntämällä Pygamen ”key” moduulia. ”Pygame.key.get_pressed” funktio palauttaa joukon ”boolean”-arvoja, jotka vastaavat jokaisen näppäimistön näppäimen tilaa. Arvo on ”True”, jos näppäintä painetaan, ja ”False”, jos näppäintä ei paineta (https://www.pygame.org/docs/ref/key.html#pygame.key.get_pressed). Tämän pohjalta on luotu ehtolauseita, joiden avulla määritellään miten pelattava hahmo reagoi mitäkin näppäintä painettaessa (kuva 7).

```
if keys[pygame.K_LEFT] and not collide_left:
    player.move_left(PLAYER_VEL)
if keys[pygame.K_RIGHT] and not collide_right:
    player.move_right(PLAYER_VEL)
```

Kuva 7.

2.1.5 Pelin päivitys

Pygame kirjaston ”Clock”- luokka tarjoaa työkalun pelin päivityksen hallintaan (https://www.pygame.org/docs/ref/time.html#pygame.time.Clock). Sitä käytetään pelissä varmistamaan, että peli päivittyy tasaisella nopeudella riippumatta koneen suorituskyvystä. Tämä toteutetaan luomalla ”Clock”- olio, jolla kutsutaan Pygamen ”tick”- metodia jokaisen pelisilmukan iteraation alussa. Metodille välitetään parametrina, kuinka monta kertaa sekunnissa pelin silmukka halutaan päivittää (https://www.pygame.org/docs/ref/time.html#pygame.time.Clock) (kuva 8).

```
def main(window):
    clock = pygame.time.Clock()
```

```
while run:

    clock.tick(FPS)
```

Kuva 8. ”Clock” olion luonti ja ”tick” metodin kutsuminen. Metodille annetaan parametriksi ”FPS” jolle on annettu arvo 60, jolla varmistetaan, että pelisilmukka päivitetään 60 kertaa sekunnissa.

2.2 Python

Pygamen moduulien ja funktioiden lisäksi koodissa hyödynnetään myös Python-ohjelmointikielen ominaisuuksia ja tekniikoita, kuten luokkia, funktioita, ehtolauseita ja silmukoita. Lähes kaikista pelin elementeistä luodaan luokka, kuten pelattava hahmosta. Funktioiden avulla luodaan erilaisia toiminnallisuuksia ja ehtolauseiden avulla määritellään milloin ja miten koodissa reagoidaan. Pelin päivitys sen ollessa käynnissä tapahtuu silmukoiden avulla. ”Main”-funktion sisällä on ”while”-silmukka, jonka jatkuu niin kauan, kun peli on käynnissä. Silmukan sisällä kutsutaan eri luokkien ”loop”- funktioita, joiden avulla niiden tilaa ja animaatioita päivitetään silmukan aikana (kuva 9).

```
run = True
while run:

    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            break
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE and player.jump_count < 2:
                player.jump()

    player.loop(FPS)
    fire.loop()
    enemy.loop()
    for fruit in fruits:
        fruit.loop()

    for fruit in fruits:
        if fruit.is_hit(player):
            player.get_points()
            fruits.remove(fruit)
            objects.remove(fruit)

    handle_move(player, objects)

    draw(window, background, bg_image, player, objects, offset_x)

    if ((player.rect.right - offset_x >= WIDTH - scroll_area_width) and player.x_vel > 0) or (
            (player.rect.left - offset_x <= scroll_area_width) and player.x_vel < 0):
        offset_x += player.x_vel

pygame.quit()
quit()
```

Kuva 9. ”Main”-funktion sisällä oleva ”while”- silmukka.

2.2.1 Luokat, yläluokat, aliluokat ja perintä

Peliini kuuluu useita luokkia, kuten pelattavaan hahmoon liittyvät luokka (Player), sekä pelin ympäristöön, esineisiin, esteisiin ja NPC-hahmoihin liittyviä luokkia (Block, Fire, Orb, Enemy). Luokkien alustus tehdään ”init”-konstruktorilla, jonka avulla voidaan määritellä mitä attribuutteja ja ominaisuuksia, joita luokka tarvitsee toimiakseen.

Luokat voivat myös periä ominaisuuksia ja metodeja toisilta luokilta ”super().__init__()” -konstruktorin avulla”. Tätä on hyödynnetty ”Object” luokan ja sen aliluokkien luomisessa. ”Object”- luokassa määritellään kaikki pelin objektien perustoimintaan liittyvät tarpeet. Lisäksi se sisältää ”draw”-metodin, joka piirtää kuvan näytölle. Välittämällä ominaisuuksia luokalta toiselle vältetään toisteisuus koodissa.

```
class Object(pygame.sprite.Sprite):
    def __init__(self, x, y, width, height, name=None):
        super().__init__()
        self.rect = pygame.Rect(x, y, width, height)
        self.image = pygame.Surface((width, height), pygame.SRCALPHA)
        self.width = width
        self.height = height
        self.name = name

    def draw(self, win, offset_x):
        win.blit(self.image, (self.rect.x - offset_x, self.rect.y))
```

Kuva 10.

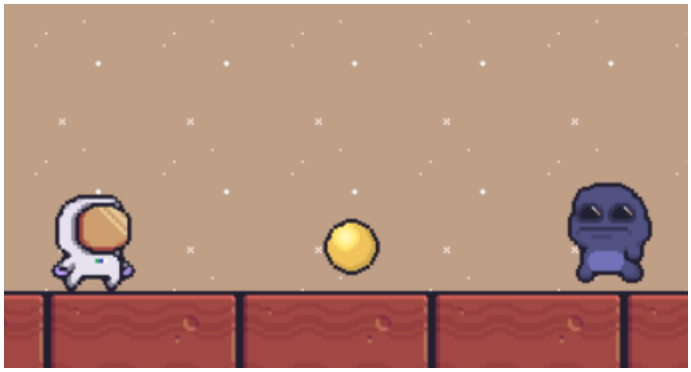
Luokat ”Object” ja ”Player” ovat puolestaan Pygamen ”Sprite”- luokan aliluokkia, ja perivät siltä ominaisuuksia ja metodeja.

3 Tulokset ja Pohdintaa

Harjoituksen myötä minulle kehittyi ymmärrystä Pygamen-kirjastosta ja erityisesti pelinkehityksessä käytettävistä toiminnoista ja luokista. Lisäksi ymmärrystä kehittyi pelihahmojen ja objektien animoinnista ja niiden hallinnasta, sekä törmäyksistä ja fysiikasta pelissä. Opin myös lisää Python-ohjelmointikielestä, sen tekniikoista ja niiden hyödyntämisestä pelinkehityksessä.

Harjoituksen jälkeen osaan alustaa pelinäkymän, luoda pelin pääsilmukan ja hallita sen päivitystaajuutta, sekä luoda peliin hahmoja ja ympäristöobjekteja käyttämällä Pygamen tarjoamia piirustus- ja animaatiotoimintoja. Pystyn myös luomaan yksinkertaisia pelielementtejä, kuten pistejärjestelmän ja hahmojen ja objektien välisiä vuorovaikutuksia.

Harjoituksen tuloksena loin yksinkertaisen version tasohyppelypelistä, jossa pelaaja voi liikkua ja hyppiä ympäristössä, kerätä pisteitä, sekä menettää ”elämiä” mikäli pelattava hahmo osuu ansaan tai viholliseen. Peli päättyy, kun pelaaja menettää kaikki elämänsä. Peli ei ole valmis versio, vaan sen oli tarkoitus toimia harjoittelun välineenä.



Linkki projektin repositoryyn: https://github.com/NotInUseHi/PygamePractice

Linkki esittelyvideoon: https://video.haaga-helia.fi/media/%22Ohjelmistokehityksen%20teknologiat%22%20-%20kurssin%20seminaarity%C3%B6/0_kjt976k1

Kiitos!

Lähdeluettelo

Pygame Frontpage, Luettavissa: https://www.pygame.org/docs/

Pygame display, Luettavissa: https://www.pygame.org/docs/ref/display.html, Luettu: 10.5.2024

Pygame image, Luettavissa: https://www.pygame.org/docs/ref/image.html). Luettu: 10.5.2024

Pygame surface, Luettavissa: https://www.pygame.org/docs/ref/surface.html). Luettu: 10.5.2024. Luettu: 10.5.2024

Pygame rect, Luettavissa: https://www.pygame.org/docs/ref/rect.html. Luettu: 10.5.2024

Pygame transform, Luettavissa: https://www.pygame.org/docs/ref/transform.html#pygame.transform.flip. Luettu: 10.5.2024

Pygame sprite, Luettavissa: https://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_mask). Luettu: 10.5.2024

Pygame mask, Luettavissa: https://www.pygame.org/docs/ref/mask.html#pygame.mask.from_surface. Luettu: 10.5.2024

Pygame key, Luettavissa: https://www.pygame.org/docs/ref/key.html#pygame.key.get_pressed). Luettu: 10.5.2024.

Pygame time, Luettavissa: https://www.pygame.org/docs/ref/time.html#pygame.time.Clock Luettu: 11.5.2024

freeCodeCamp.org 20.3.2023. Python Platformer Game Tutorial for Beginners. Video. Katsottavissa: https://www.youtube.com/watch?v=6gLeplbqtqg. Katsottu: 10.5.2024.