CS 11 Machine Problem 2

“RACECAR”

Programmer’s Guide

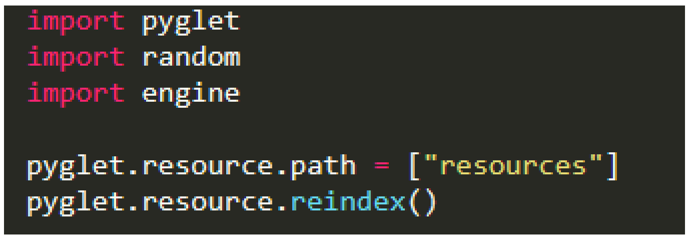
Ashley Dancel De Castro

Justin Nicolas Fabian

Ayla Nikki Lorreen Odasco

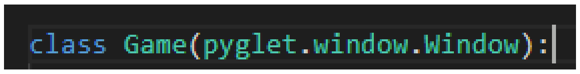
The Race Car is an obstacle-dodging game that consists of a car and different rocks to dodge. The game starts when the player presses “Enter” and constantly tracks the high score of the player while updating the current score. The high score is saved in a text file for future reference.

For the program code, we have the main.py, engine.py, and interface.py. The game files such as the images/sprites, background, sound effects, and the text file responsible for storing the high score in-game are all stored in the resources folder.



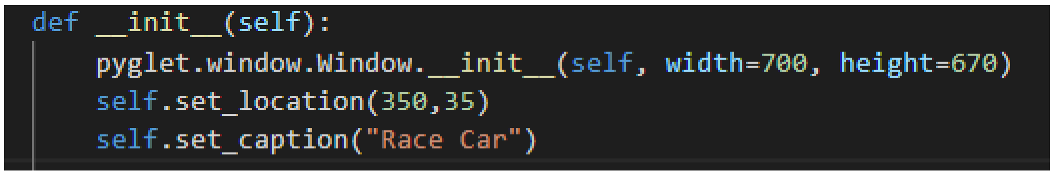
```python
import pyglet
import random
import engine

pyglet.resource.path = ["resources"]
pyglet.resource.reindex()
```

In this code, it imports different modules namely pyglet, random, and engine. The pyglet file resource is set to the “resources” folder.



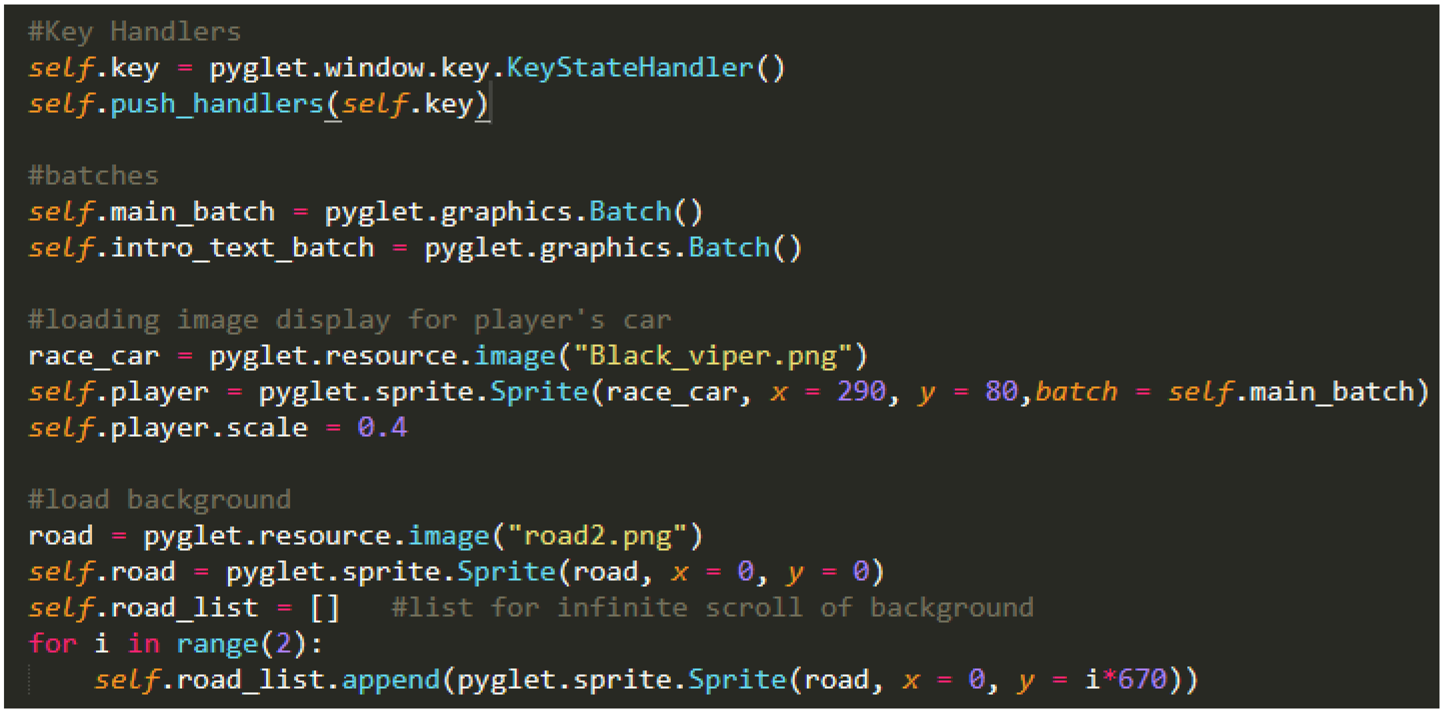
```python
class Game(pyglet.window.Window):
```

This code is responsible for creating the Game() object, which will be used to start the game.



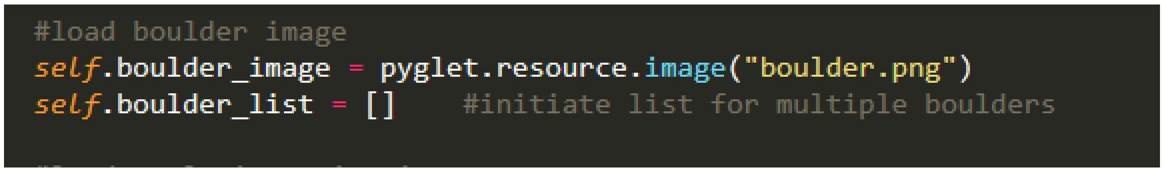
```python
def __init__(self):
    pyglet.window.Window.__init__(self, width=700, height=670)
    self.set_location(350,35)
    self.set_caption("Race Car")
```

The __init__ method is basically acting as a constructor for the game. The self variable is just a representation of the object itself. We are setting the window size to width 700 and height 670. The location on which the window appears is set on (350,35) and the title for the game is “Race Car”.



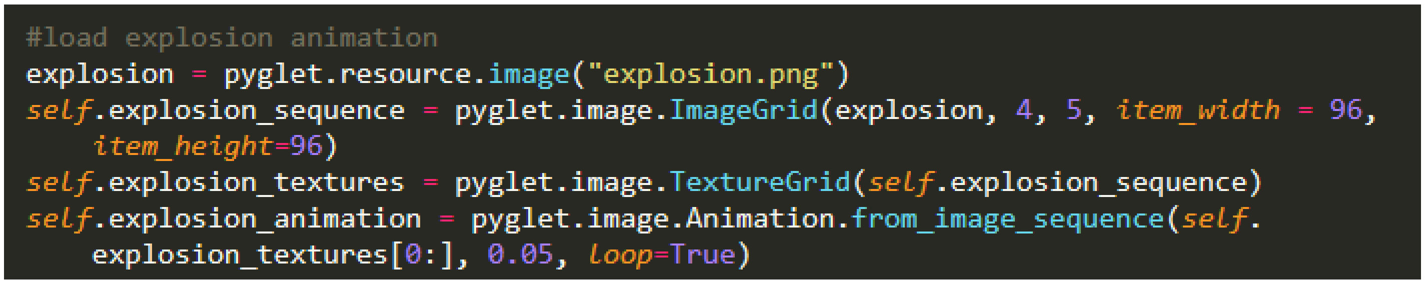
```python
#Key Handlers
self.key = pyglet.window.key.KeyStateHandler()
self.push_handlers(self.key)

#batches
self.main_batch = pyglet.graphics.Batch()
self.intro_text_batch = pyglet.graphics.Batch()

#loading image display for player's car
race_car = pyglet.resource.image("Black_viper.png")
self.player = pyglet.sprite.Sprite(race_car, x = 290, y = 80,batch = self.main_batch)
self.player.scale = 0.4

#load background
road = pyglet.resource.image("road2.png")
self.road = pyglet.sprite.Sprite(road, x = 0, y = 0)
self.road_list = []   #list for infinite scroll of background
for i in range(2):
    self.road_list.append(pyglet.sprite.Sprite(road, x = 0, y = i*670))
```

Here is just an continuation from the previous code displayed. The KeyStateHandler is just for storing the keyboard state, which will be used later. The next set of code is the Batch, which loads all the sprites altogether, it renders different vertex list in a single draw call. Then, the next one is for loading the sprite for the race car/player. It sets the position of the sprite from the beginning and the scale of the sprite to fit into the game. The last one consists of codes that will load the background and we will scroll it endlessly as long as the application runs. This makes the car look like it is moving forward.



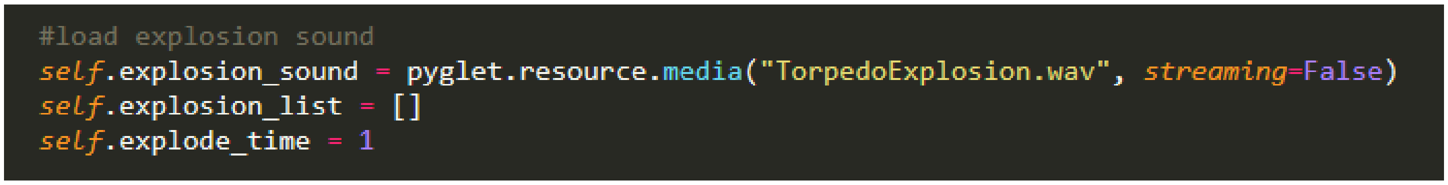
```python
#load boulder image
self.boulder_image = pyglet.resource.image("boulder.png")
self.boulder_list = []    #initiate list for multiple boulders
```

This loads the image file we have in the resources folder. This is basically what decides what the boulders on the road will look like. We also initialized a list for multiple boulders on the road.



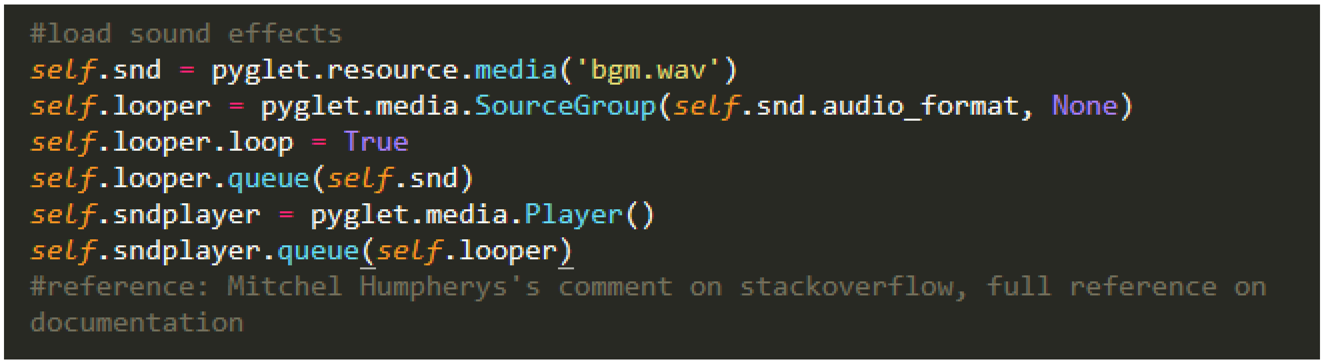
```python
#load explosion animation
explosion = pyglet.resource.image("explosion.png")
self.explosion_sequence = pyglet.image.ImageGrid(explosion, 4, 5, item_width = 96,
    item_height=96)
self.explosion_textures = pyglet.image.TextureGrid(self.explosion_sequence)
self.explosion_animation = pyglet.image.Animation.from_image_sequence(self.
    explosion_textures[0:], 0.05, loop=True)
```

This code is for accessing the texture grid as a single texture. It basically loads up the sprites “explosion.png” to be used as the animation of the explosion.



```python
#load explosion sound
self.explosion_sound = pyglet.resource.media("TorpedoExplosion.wav", streaming=False)
self.explosion_list = []
self.explode_time = 1
```

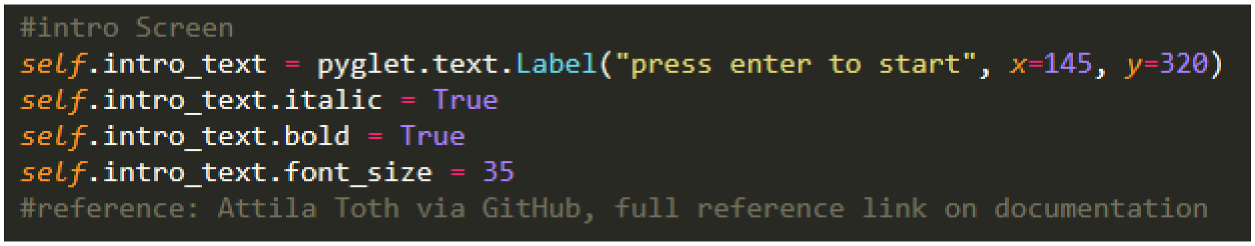
Along with the previous code, we need to accompany it with the an explosion sound using the “TorpedoExplosion.wav”, I know it’s a race car and it isn’t supposed to have a torpedo but it’s the sound that counts.



```python
#load sound effects
self.snd = pyglet.resource.media('bgm.wav')
self.looper = pyglet.media.SourceGroup(self.snd.audio_format, None)
self.looper.loop = True
self.looper.queue(self.snd)
self.sndplayer = pyglet.media.Player()
self.sndplayer.queue(self.looper)
#reference: Mitchel Humpherys's comment on stackoverflow, full reference on
documentation
```

We used this set of codes for the loop of the background music of the game. The looper is set to True which is then obviously will loop the music.

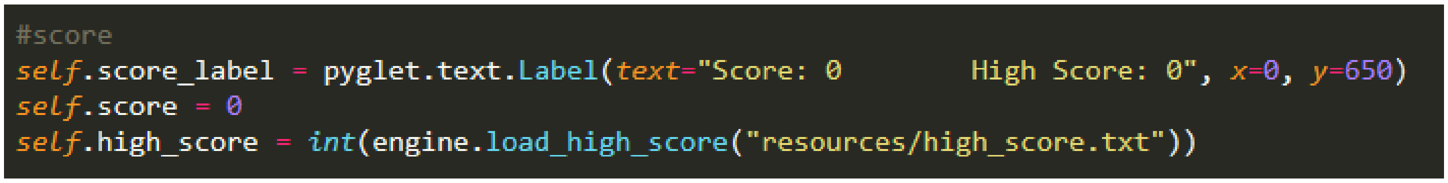
See more: https://stackoverflow.com/questions/27391240/how-to-play-music-continuously-in-pyglet?fbclid=IwAR3OSQeoLCeRr_ikisI9pmFcqKZ8XKzaNT2GB5Aqpk2kbTEcTwy4Bzb9NjQ



```python
#intro Screen
self.intro_text = pyglet.text.Label("press enter to start", x=145, y=320)
self.intro_text.italic = True
self.intro_text.bold = True
self.intro_text.font_size = 35
#reference: Attila Toth via GitHub, full reference link on documentation
```

This is just the intro screen, asking the player to press Enter to start the game.

See more: https://github.com/totex/Space-Invaders-in-Pyglet/blob/master/Space%20Invaders%201.0%20prototype/main.py



```python
#score
self.score_label = pyglet.text.Label(text="Score: 0        High Score: 0", x=0, y=650)
self.score = 0
self.high_score = int(engine.load_high_score("resources/high_score.txt"))
```

This displays the current score of the player along with the current high score of the game. This is where the text file comes in, because this is where we save the high score.

```
#game over
self.game_over_label = pyglet.text.Label(text="Game Over!", x=350, y=330)
self.game_over_label.anchor_x = "center"
self.game_over_label.anchor_y = "center"
self.game_over_label.font_name = "Courier New"
self.game_over_label.font_size = 50
```

This code sets the Game Over screen if the player hits a boulder along the way.

```
#retry
self.retry_label = pyglet.text.Label(text="Retry Game? \nPress Y to retry, else N.", x
    =350, y=240)
self.retry_label.anchor_x = 'center'
self.retry_label.anchor_y = 'center'
self.retry_label.font_name = "Arial"
self.retry_label.font_size = 20
```

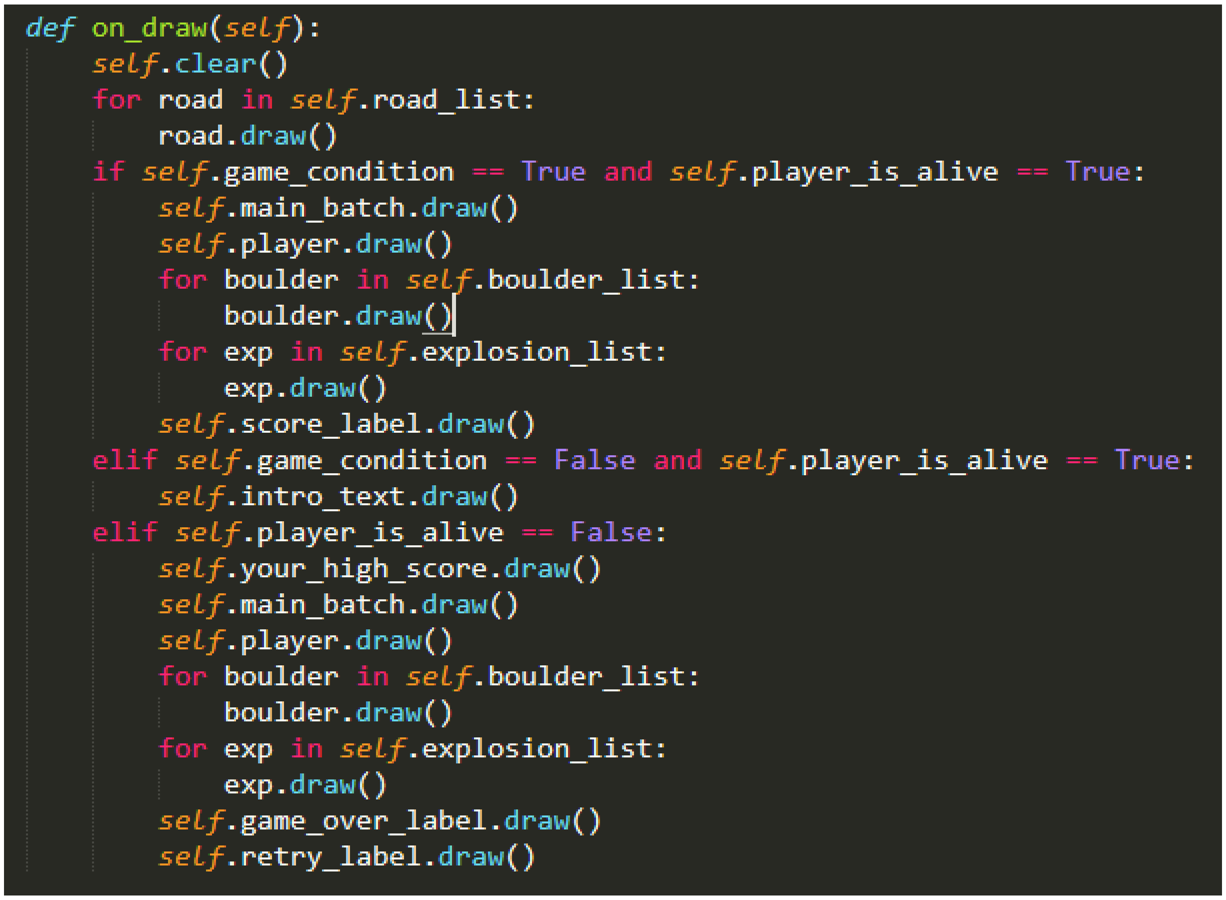
This displays the question if the player is going to retry the game or close the game. Prompting Y to retry and N if the player chooses to exit.

```
#score if got high score
self.your_high_score = pyglet.text.Label("", x=350, y=200)
self.your_high_score.anchor_x = 'center'
self.your_high_score.anchor_y = 'center'
self.your_high_score.font_name = "Arial"
self.your_high_score.font_size = 18

#game conditions
self.player_is_alive = True
self.game_condition = False
```

This will set an empty string for the score of the player, and we have initialize a boolean value for the game conditions.

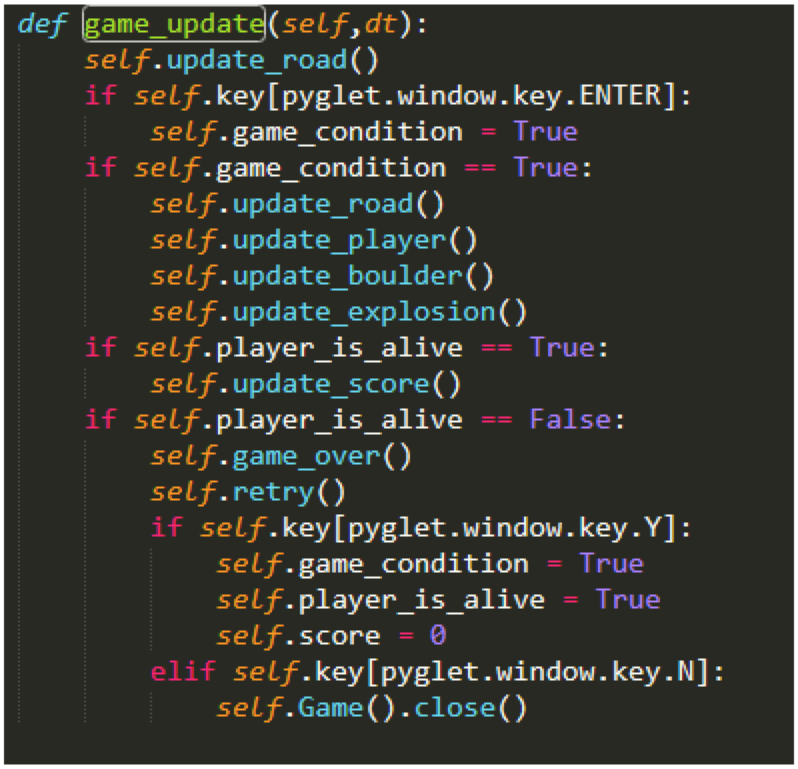
```
pyglet.clock.schedule_interval(self.game_update, 1/60.0)
```

This is just to update the function every second that passes.



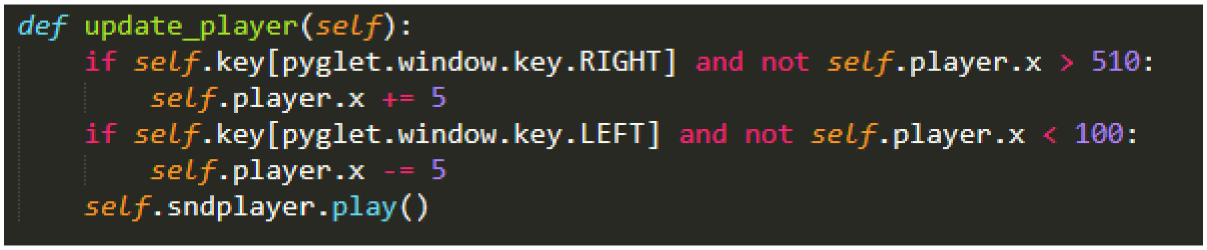
```python
def on_draw(self):
    self.clear()
    for road in self.road_list:
        road.draw()
    if self.game_condition == True and self.player_is_alive == True:
        self.main_batch.draw()
        self.player.draw()
        for boulder in self.boulder_list:
            boulder.draw()
        for exp in self.explosion_list:
            exp.draw()
        self.score_label.draw()
    elif self.game_condition == False and self.player_is_alive == True:
        self.intro_text.draw()
    elif self.player_is_alive == False:
        self.your_high_score.draw()
        self.main_batch.draw()
        self.player.draw()
        for boulder in self.boulder_list:
            boulder.draw()
        for exp in self.explosion_list:
            exp.draw()
        self.game_over_label.draw()
        self.retry_label.draw()
```

This function on_draw is to decide what should appear on the screen according to certain conditions. It includes displaying the road, the in-game conditions, the intro, as well as when the race car gets hit with a boulder. At the end of the code, we also see the game over display screen along with the retry screen.



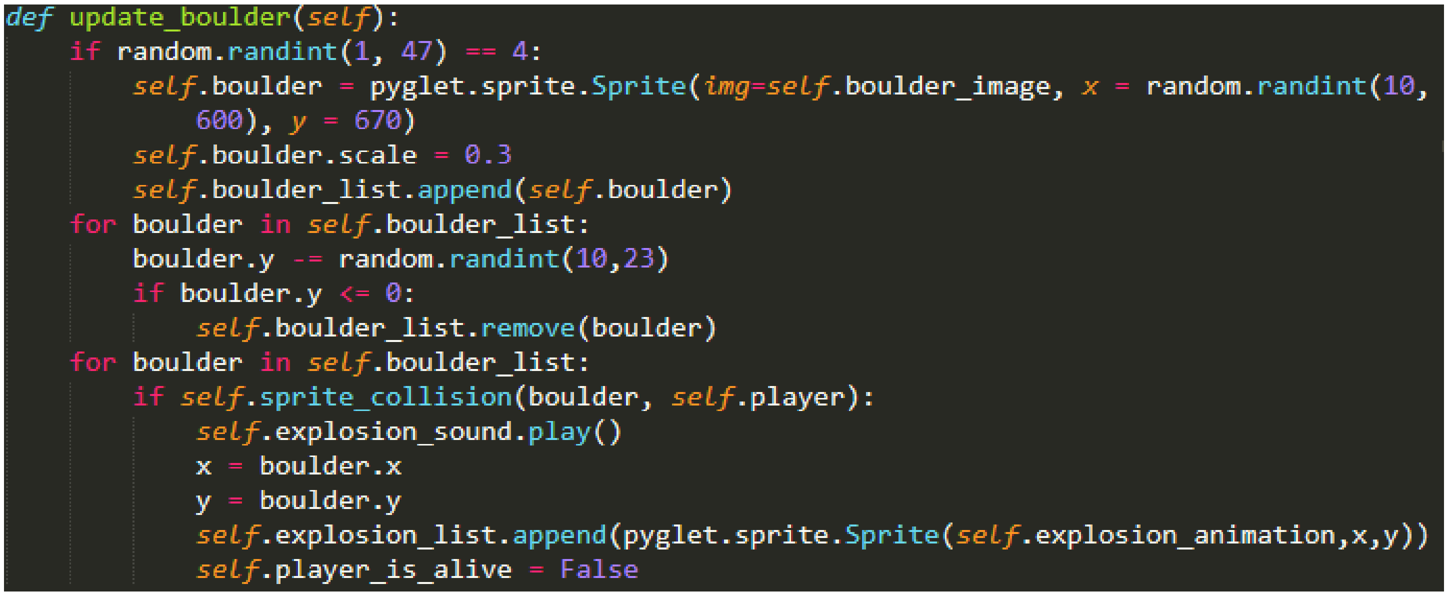
```
def game_update(self,dt):
    self.update_road()
    if self.key[pyglet.window.key.ENTER]:
        self.game_condition = True
    if self.game_condition == True:
        self.update_road()
        self.update_player()
        self.update_boulder()
        self.update_explosion()
    if self.player_is_alive == True:
        self.update_score()
    if self.player_is_alive == False:
        self.game_over()
        self.retry()
        if self.key[pyglet.window.key.Y]:
            self.game_condition = True
            self.player_is_alive = True
            self.score = 0
        elif self.key[pyglet.window.key.N]:
            self.Game().close()
```

The game_update function is for the updating of the game in real time while the application is running. It also decides what will happen when specific keys on the keyboard are pressed. When the game is ongoing, it updates the score, road, player, boulder and the explosion.



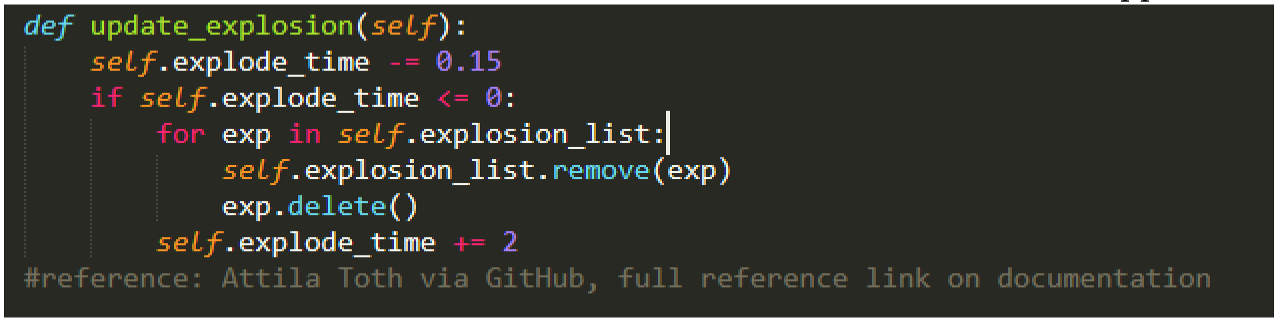
```
def update_player(self):
    if self.key[pyglet.window.key.RIGHT] and not self.player.x > 510:
        self.player.x += 5
    if self.key[pyglet.window.key.LEFT] and not self.player.x < 100:
        self.player.x -= 5
    self.sndplayer.play()
```

The update_player function is for the movement of the race car. This is a side-scrolling game and we need to specify the certain range where the car can move using the Left and Right arrow keys.



```python
def update_boulder(self):
    if random.randint(1, 47) == 4:
        self.boulder = pyglet.sprite.Sprite(img=self.boulder_image, x = random.randint(10,
            600), y = 670)
        self.boulder.scale = 0.3
        self.boulder_list.append(self.boulder)
    for boulder in self.boulder_list:
        boulder.y -= random.randint(10,23)
        if boulder.y <= 0:
            self.boulder_list.remove(boulder)
    for boulder in self.boulder_list:
        if self.sprite_collision(boulder, self.player):
            self.explosion_sound.play()
            x = boulder.x
            y = boulder.y
            self.explosion_list.append(pyglet.sprite.Sprite(self.explosion_animation,x,y))
            self.player_is_alive = False
```

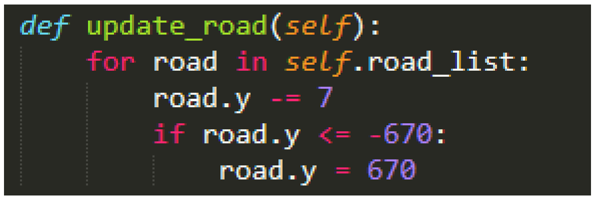
The update_boulder function is for checking if the player collides with the race car on the screen. This also sets random coordinates where the boulders will appear.



```python
def update_explosion(self):
    self.explode_time -= 0.15
    if self.explode_time <= 0:
        for exp in self.explosion_list:
            self.explosion_list.remove(exp)
            exp.delete()
        self.explode_time += 2
#reference: Attila Toth via GitHub, full reference link on documentation
```

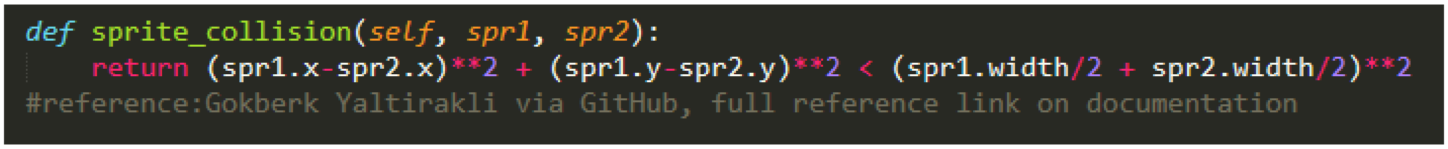
The update_explosion function just updates the explosion and animates the explosion to make it look more realistic.

See more: https://github.com/totex/Space-Invaders-in-Pyglet/blob/master/Space%20Invaders%201.0%20prototype/main.py



```python
def update_road(self):
    for road in self.road_list:
        road.y -= 7
        if road.y <= -670:
            road.y = 670
```

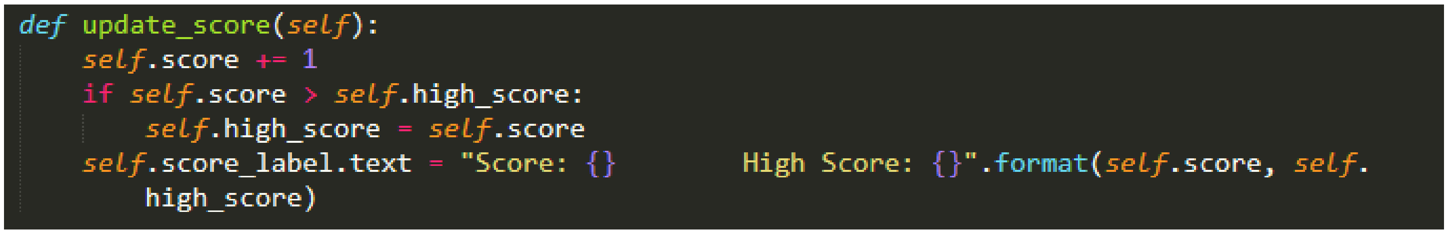
The update_road function is responsible for making it look like that the road is scrolling endlessly.



```python
def sprite_collision(self, spr1, spr2):
    return (spr1.x-spr2.x)**2 + (spr1.y-spr2.y)**2 < (spr1.width/2 + spr2.width/2)**2
#reference:Gokberk Yaltirakli via GitHub, full reference link on documentation
```

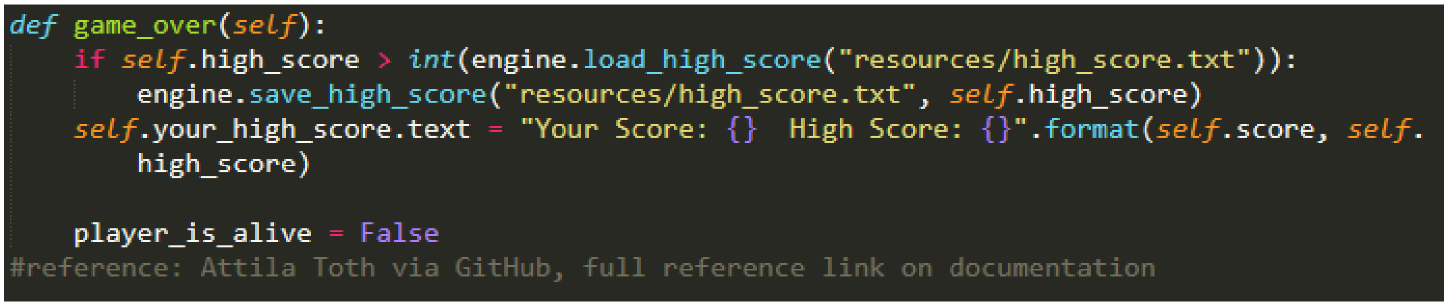
The sprite_collision code is used for checking if the sprites collide with each other, in this case, the boulders and the race car.

See more:
https://github.com/gkbrk/AsteroidRacer?fbclid=IwAR09KGXKzuPAbQeREN5yaxxDQ2WYKxSQq1g29u328_3Aril12WGc6rjWTqI



```
def update_score(self):
    self.score += 1
    if self.score > self.high_score:
        self.high_score = self.score
    self.score_label.text = "Score: {}        High Score: {}".format(self.score, self.
        high_score)
```

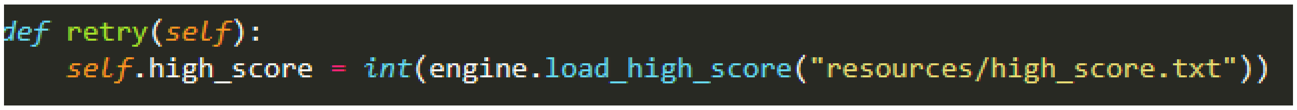
The update_score function’s purpose is to update the score being showed in the screen in order for the player to be informed what the current score and high score is.



```
def game_over(self):
    if self.high_score > int(engine.load_high_score("resources/high_score.txt")):
        engine.save_high_score("resources/high_score.txt", self.high_score)
    self.your_high_score.text = "Your Score: {}  High Score: {}".format(self.score, self.
        high_score)

    player_is_alive = False
#reference: Attila Toth via GitHub, full reference link on documentation
```

The game_over function is for saving the new high score if ever the player beats the previous record and then displays the current score along with the high score. It sets the player_is_alive to False.

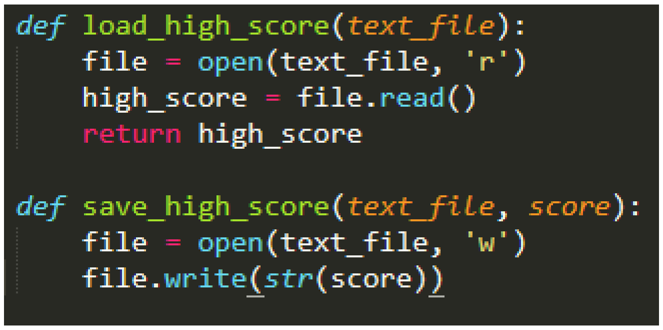
See more: https://github.com/totex/Space-Invaders-in-Pyglet/blob/master/Space%20Invaders%201.0%20prototype/main.py



```
def retry(self):
    self.high_score = int(engine.load_high_score("resources/high_score.txt"))
```

The retry function is basically setting the high score with the data from the text file.

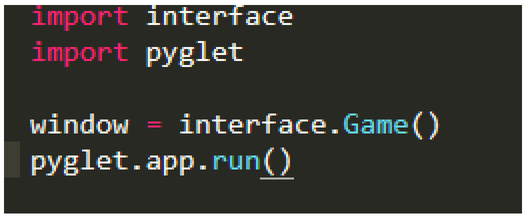
The engine.py contains the following codes:



```
def load_high_score(text_file):
    file = open(text_file, 'r')
    high_score = file.read()
    return high_score

def save_high_score(text_file, score):
    file = open(text_file, 'w')
    file.write(str(score))
```

We’re going to load the high score from file using the load_high_score function and also save the high score in a text file using the save_high_score function.

The main.py contains the following codes:



```
import interface
import pyglet

window = interface.Game()
pyglet.app.run()
```

It imports the interface and the pyglet module. This is where we run our game.

References:

"Free Sound Effects," [Online]. Available: https://www.freesoundeffects.com/free-sounds/explosion-10070/. [Accessed 30 November 2018].

Alucard, "2D Top Down Endless Runner Background," 2016. [Online]. Available: http://devsupply.blogspot.com/. [Accessed 30 November 2018].

A.Toth, "Space Invaders Pyglet," GitHub, 14 October 2017. [Online]. Available: https://github.com/totex/Space-Invaders-in-Pyglet/blob/master/Space%20Invaders%201.0%20prototype/main.py. [Accessed 30 November 2018].

Cuzco, "Explosion," Open Game Art, 2 December 2010. [Online]. Available: https://opengameart.org/content/explosion. [Accessed 30 November 2018].

G. Yaltirakli, "Asteroid Racer," GitHub, 8 March 2014. [Online]. Available: https://github.com/gkbrk/AsteroidRacer?fbclid=IwAR09KGXKzuPAbQeREN5yaxxDQ2WYKxSQq1g29u328_3Aril12WGc6rjWTqI. [Accessed 30 November 2018].

M. Humpherys, "How to play music continuously in pyglet," StackOverflow, 29 October 2018. [Online]. Available: https://stackoverflow.com/questions/27391240/how-to-play-music-continuously-in-pyglet. [Accessed 3 December 2018].

R. Denny, "Free Boulder Pack," Asset Store, 2017. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/landscapes/free-boulder-pack-68611. [Accessed 30 November 2018].

S. Yadav, ""Top Down Vehicle Sprite Pack"," Unlucky Studio, 2015. [Online]. Available: http://unluckystudio.com/. [Accessed 30 November 2018].