



Assignment Cover Letter

(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	Ihsan Kamal	Fauzan	2101720700

Course Code	: COMP6502	Course Name	: Introduction to Programming
--------------------	------------	--------------------	-------------------------------

Class	: L1BC	Name of Lecturer(s)	: 1. Minaldi Loeis 2. Jude Martinez
--------------	--------	----------------------------	--

Major	: Computer Science
--------------	--------------------

Title of Assignment (if any)	: Space Battle
--	----------------

Type of Assignment	: Final Project
---------------------------	-----------------

Submission Pattern

Due Date	: 8-11-2017	Submission Date	: 8-11-2017
-----------------	-------------	------------------------	-------------

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to Binus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Fauzan Ihsan Kamal

“Space Battle”

Name = Fauzan Ihsan Kamal

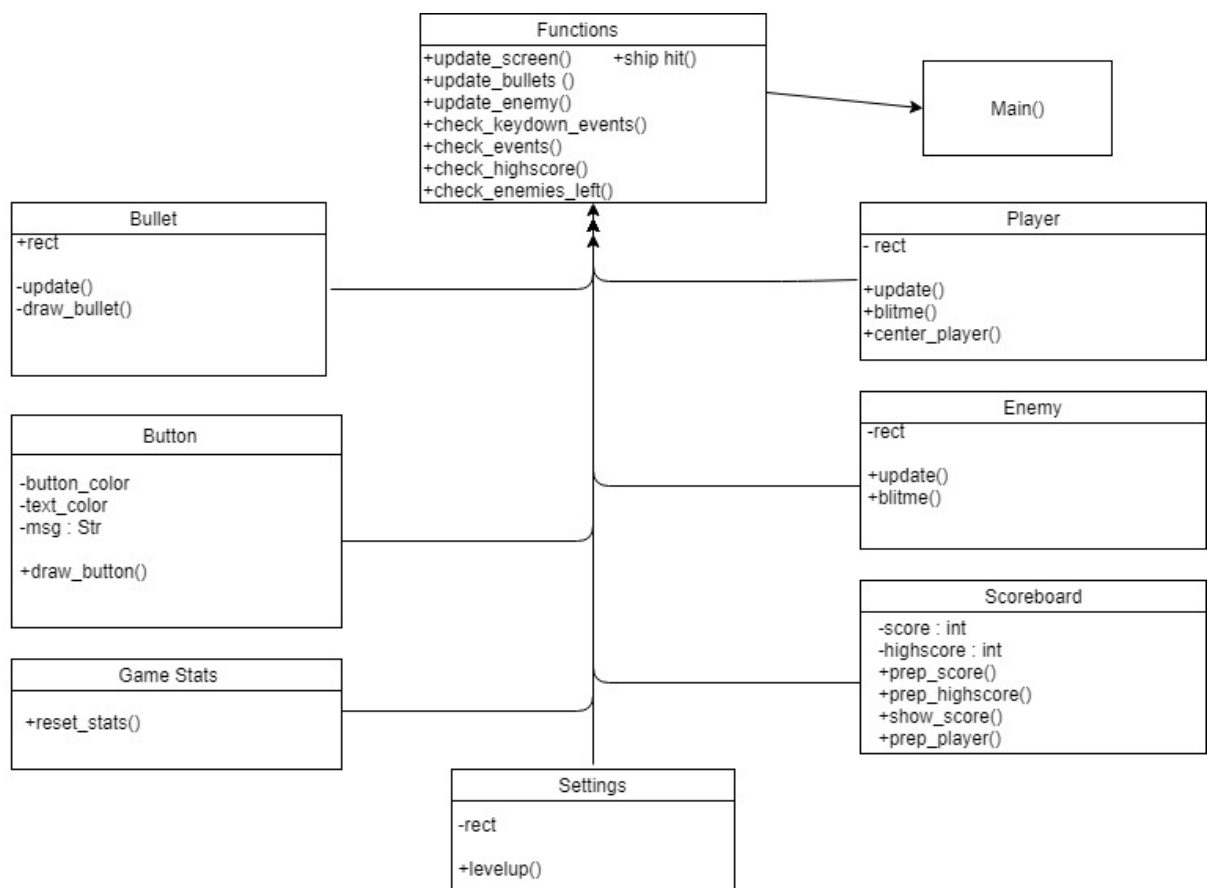
ID = 2101720700

I. Description

The function of this program :

This program is meant for the people who are bored and want to play a 2D endless shooter game on their computer. This game takes place in space where the player must shoot the aliens as much as they can to survive and gain scores and the difficulty is based on the player’s score. The higher the score, the more difficult the game gets. The game ends when an alien hits the player’s ship or reaches the edge of the left screen. This game is based on alien invasion game from python crash course.

II. UML Diagram



Text

III. Explanation of each function

Class Player (*player.py*)

- The purpose of this class is to adjust the player's position and also the player's movement.
- In *def update()*, I adjust the player's movement and also limits the movement. So the player can't get pass through the edge of the screen.
- In *def center_player()*, I input the player's position when the game starts or when the player respawns.
- *def blime* is to draw the player's ship into the screen

```
4 class Player(Sprite):
5     def __init__(self, ai_settings, screen):
6         super(Player, self).__init__()
7         self.screen = screen
8         self.ai_settings = ai_settings
9
10        self.image = pygame.image.load('images/futuramaship.png')
11        self.rect = self.image.get_rect()
12        self.screen_rect = screen.get_rect()
13
14
15        self.rect.centery = self.screen_rect.centery
16        self.rect.left = self.screen_rect.left
17        self.center = float(self.rect.centery)
18
19
20        self.moving_down = False
21        self.moving_up = False
22        self.moving_right = False
23        self.moving_left = False
24
25
26    def update(self):
27        if self.moving_down and self.center <= self.screen_rect.bottom:
28            self.center += self.ai_settings.player_speed_factor
29
30        if self.moving_up and self.center >= self.screen_rect.top:
31            self.center -= self.ai_settings.player_speed_factor
32
33        if self.moving_right and self.rect.right <= self.screen_rect.right:
34            self.rect.left += self.ai_settings.player_speed_factor
35
36        self.rect.centery = self.center
```

Player > __init__()

```
21        self.moving_up = False
22        self.moving_right = False
23        self.moving_left = False
24
25
26    def update(self):
27        if self.moving_down and self.center <= self.screen_rect.bottom:
28            self.center += self.ai_settings.player_speed_factor
29
30        if self.moving_up and self.center >= self.screen_rect.top:
31            self.center -= self.ai_settings.player_speed_factor
32
33        if self.moving_right and self.rect.right <= self.screen_rect.right:
34            self.rect.left += self.ai_settings.player_speed_factor
35
36        if self.moving_left and self.rect.left >= self.screen_rect.left:
37            self.rect.left -= self.ai_settings.player_speed_factor
38
39        self.rect.centery = self.center
40
41
42
43    def blime(self):
44        self.screen.blit(self.image, self.rect)
45
46    def center_player(self):
47        self.center = self.screen_rect.centery
48        self.rect.left = self.screen_rect.left
49
50
51
```

Player > __init__()

Class Enemy (*enemy.py*)

- The purpose of this class is to adjust the enemy's position when it spawns and to set the enemy's direction when it moves
- For the position, I use `self.rect.y = random.randint(0,500)` to make the enemy spawn randomly in the y position between 0 and 500.
- In `def update()`, I set it `--` so that the enemy can move to the left from the edge of the right screen.
- `def blitme()` is to draw the enemy

```
4
5 class Enemy(Sprite):
6     """a class to represent a single enemy"""
7     def __init__(self, ai_settings, screen):
8         super(Enemy, self).__init__()
9         self.screen = screen
10        self.ai_settings = ai_settings
11
12
13        self.image = pygame.image.load('images/cartoon-spaceship1.png')
14        self.rect = self.image.get_rect()
15        self.rect.right = 1300
16
17
18        self.rect.y = random.randint(0,500)
19
20        self.x = float(self.rect.x)
21        self.y = float(self.rect.y)
22
23        #to draw the enemy
24        def blitme(self):
25            self.screen.blit(self.image, self.rect)
26
27        #to set the direction of enemy
28        def update(self):
29            self.rect.right -= self.ai_settings.enemy_speed_factor + self.ai_settings.level
30
31
32
33
34
```

Enemy > blitme()

Class Bullet (*bullet.py*)

- This class is to adjust the position and the direction of the player's bullet.
- In `def update()`, I adjust the speed and the position of the bullet
- `def draw_bullet` is to draw the bullet into the screen

```
1 import pygame
2 from pygame.sprite import Sprite
3
4 class Bullet(Sprite):
5     def __init__(self, ai_settings, screen, player):
6         super(Bullet, self).__init__()
7         self.screen = screen
8
9         self.rect = pygame.Rect(0,0,ai_settings.bullet_width, ai_settings.bullet_height)
10        self.rect.centerx = player.rect.centerx
11        self.rect.centery = player.rect.centery
12
13        self.x = float(self.rect.x)
14        self.color = ai_settings.bullet_color
15
16
17        def update(self):
18            self.x += 10
19            self.rect.x = self.x
20
21
22        def draw_bullet(self):
23            pygame.draw.rect(self.screen, self.color, self.rect)
24
25
26
27
```

Bullet

Class Settings (*settings.py*)

- This class stores all of the game's settings (screen, level, player, enemy, etc) so I it makes the call code simpler and I can modify the settings easily.
- *def levelup* is to set the score on when the level changes

```
1 class Settings():
2     def __init__(self):
3         #Screen settings
4         self.screen_width = 1200
5         self.screen_height = 600
6
7         #level settings
8         self.level = 1
9
10        #player settings
11        self.player_speed_factor = 7
12        self.ship_limit = 0
13
14        #enemy settings
15        self.enemy_speed_factor = 7
16
17        #bullet settings
18        self.bullet_width = 16
19        self.bullet_height = 3
20        self.bullet_color = 255,0,0
21        self.bullets_allowed = 20
22
23        #scoring
24        self.enemy_points = 50
25
26        #function to level up each time player gain +500 score
27        def levelup(self, score):
28            if score > 500:
29                self.level = 2
30            if score > 1000:
31                self.level = 3
```

Settings > __init__()

Class Button (*button.py*)

- The usage of this class is to make the play button appear before starting the game.
- *def prep_msg* is to turn the text into a rendered image and to center the text on the button.
- *def draw_button* is to draw the play button into the screen

```
1 import pygame.font
2
3 #class to make the play button
4 class Button():
5     def __init__(self, ai_settings, screen, msg):
6         self.screen = screen
7         self.screen_rect = screen.get_rect()
8
9         self.width, self.height = 200, 50
10        self.button_color = (255,0, 0)
11        self.text_color = (255, 255, 255)
12        self.font = pygame.font.SysFont(None, 48)
13
14        self.rect = pygame.Rect(0, 0, self.width, self.height)
15        self.rect.center = self.screen_rect.center
16
17        self.prep_msg(msg)
18
19    def prep_msg(self, msg):
20        self.msg_image = self.font.render(msg, True, self.text_color, self.button_color)
21        self.msg_image_rect = self.msg_image.get_rect()
22        self.msg_image_rect.center = self.rect.center
23
24    def draw_button(self):
25        self.screen.fill(self.button_color, self.rect)
26        self.screen.blit(self.msg_image, self.msg_image_rect)
```

Class Scoreboard (scoreboard.py)

- The purpose of this class is to make the score and the highscore appear in-game and also to adjust position of the score and the highscore.
- In *def prep_score*, I adjust the score's position at the top right of the screen and I also add *score_str* to add a coma when the score reach ≥ 1000 .
- *def prep_high_score* is the same as *def prep_score* but this function adjust the highscore.
- *def show* is to draw the score into the screen

```
7 class Scoreboard():
8     def __init__(self, ai_settings, screen, stats):
9         self.screen = screen
10        self.screen_rect = screen.get_rect()
11        self.stats = stats
12        self.ai_settings = ai_settings
13
14        #font settings for scoring
15        self.text_color = (250,250,250)
16        self.font = pygame.font.SysFont(None, 30)
17
18        self.prep_score()
19        self.prep_high_score()
20        self.prep_player()
21
22        #turn the score into a rendered image
23        def prep_score(self):
24            rounded_score = int(round(self.stats.score, -1))
25            score_str = "{:,}".format(rounded_score)
26            self.score_image = self.font.render(score_str, True, self.text_color)
27            self.score_rect = self.score_image.get_rect()
28            self.score_rect.right = self.screen_rect.right - 20
29            self.score_rect.top = 20
30            print(self.stats.score)
31
32        #draw the score to the screen
33        def show_score(self):
34            self.screen.blit(self.score_image, self.score_rect)
35            self.screen.blit(self.high_score_image, self.high_score_rect)
36            self.players.draw(self.screen)
37
```

Scoreboard > __init__()

```
29         self.score_rect.top = 20
30         print(self.stats.score)
31
32        #draw the score to the screen
33        def show_score(self):
34            self.screen.blit(self.score_image, self.score_rect)
35            self.screen.blit(self.high_score_image, self.high_score_rect)
36            self.players.draw(self.screen)
37
38        def prep_high_score(self):
39            high_score = int(round(self.stats.high_score, -1))
40            high_score_str = "{:,}".format(high_score)
41            self.high_score_image = self.font.render(high_score_str, True, self.text_color)
42
43            #Center the high score at the top of the screen
44            self.high_score_rect = self.high_score_image.get_rect()
45            self.high_score_rect.centerx = self.screen_rect.centerx
46            self.high_score_rect.top = self.score_rect.top
47
48        def prep_player(self):
49            self.players = Group()
50            for player_number in range(self.stats.ships_left):
51                player = Player(self.ai_settings, self.screen)
52                player.rect.x = -20
53                player.rect.y = -20
54                self.players.add(player)
55
56
57
58
59
```

Scoreboard > __init__()

Class Game Stats (*game_stats.py*)

- This class is to set the game statistics such as the player's life limit and to reset the stats when the game restarts

```
1  class GameStats():
2
3      def __init__(self, ai_settings):
4          self.ai_settings = ai_settings
5          self.reset_stats()
6          self.game_active = True
7          self.high_score = 0
8
9          self.game_active = False
10
11     def reset_stats(self):
12         self.ships_left = self.ai_settings.ship_limit
13         self.score = 0
14
15
16
```

Functions (*functions.py*)

- *def check_events()* :
The usage of this function is to manage events so it can respond to a keypresses and mouse events.
- *def play_button()*:

This function is to check if the play button whether it's inactive when the program first start.

- *def check_keydown_events()*:

This function is to check any key push input. I also input a sound effect on the spacebar key using *pygame.mixer*, so it will make a noise everytime the player shoot a laser.

- *def check_update_bullets()*:

This function is to update the bullets position and to run *check_bullets_collision*.

- *def check_bullet_collision():*

This function is to create a collision events and its response and also it will update the score whenever the bullet hits an enemy.

- *def update_enemy():*

This function is to update the enemy against collision with the player or against the left edge of the screen.

- *def ship_hit():*

This function is to reset the player's position when the enemy hits the player and it will also update the stats.

- *def check_highscore():*

This function is to check if there is a new highscore and it will update the highscore on the top of the screen.

- *def check_enemy_left():*

This function is to check if the enemies have reached the left edge of the screen and if it's true, it will run *def ship_hit*.

- *def update_screen():*

This function is to update all the events that is happening on the screen such as the player's movement, the bullets, spawning the enemy, etc.

```

1  import sys
2
3  import pygame,time
4
5  from bullet import Bullet
6  from pygame.sprite import *
7  from pygame import *
8  from enemy import Enemy
9  from time import sleep
10
11
12
13  def check_events(ai_settings,screen,stats,sb, play_button, _player,enemies,bullets):
14      for event in pygame.event.get():
15          if event.type == pygame.QUIT:
16              sys.exit()
17
18          elif event.type == pygame.MOUSEBUTTONDOWN:
19              mouse_x, mouse_y = pygame.mouse.get_pos()
20              check_play_button(ai_settings, screen, stats,sb, play_button, player, enemies, bullets, mouse_x, mouse_y)
21
22
23          elif event.type == pygame.KEYDOWN:
24              check_keydown_events(event,ai_settings,screen,player,bullets)
25              if event.key == pygame.K_DOWN:
26                  player.moving_down = True
27              elif event.key == pygame.K_UP:
28                  player.moving_up = True
29              elif event.key == pygame.K_LEFT:
30                  player.moving_left = True
31              elif event.key == pygame.K_RIGHT:

```

check_enemies_l...


```

34     elif event.type == pygame.KEYUP:
35         if event.key == pygame.K_DOWN:
36             player.moving_down = False
37         if event.key == pygame.K_UP:
38             player.moving_up = False
39         elif event.key == pygame.K_LEFT:
40             player.moving_left = False
41         elif event.key == pygame.K_RIGHT:
42             player.moving_right = False
43
44     #function to check the play button when the program first start
45     def check_play_button(ai_settings, screen, stats, sb, play_button, player, enemies, bullets, mouse_x, mouse_y):
46         #to reset the game each time user click the play button
47         button_clicked = play_button.rect.collidepoint(mouse_x, mouse_y)
48         if button_clicked and not stats.game_active:
49             stats.game_active = True
50             stats.reset_stats()
51             pygame.mouse.set_visible(False)
52             enemies.empty()
53             bullets.empty()
54             sb.prep_score()
55             sb.prep_high_score()
56             sb.prep_player()
57
58
59
60         player.center_player()
61
62     def check_keydown_events(event, ai_settings, screen, player, bullets):
63         if event.key == pygame.K_SPACE:
64             if len(bullets) < ai_settings.bullets_allowed:

```

check_enemies_l...

```

61
62     def check_keydown_events(event, ai_settings, screen, player, bullets):
63         if event.key == pygame.K_SPACE:
64             if len(bullets) < ai_settings.bullets_allowed:
65                 pygame.mixer.pre_init(44100, -16, 2, 2048)
66                 pygame.mixer.music.load('sound/Laser Blasts-SoundBible.com-108608437.mp3')
67                 pygame.mixer.music.set_volume(0.1)
68                 pygame.mixer.music.play(0)
69                 new_bullet = Bullet(ai_settings, screen, player)
70                 bullets.add(new_bullet)
71
72     #function to update the bullets position and the collisions
73     def update_bullets(ai_settings, screen, stats, sb, player, enemies, bullets):
74         bullets.update()
75         check_bullet_enemy_collisions(ai_settings, screen, stats, sb, player, enemies, bullets)
76
77     #to check if the bullet hit the enemy ship
78     def check_bullet_enemy_collisions(ai_settings, screen, stats, sb, player, enemies, bullets):
79         collisions = pygame.sprite.groupcollide(bullets, enemies, True, True)
80         for bullet in bullets.copy():
81             if bullet.rect.bottom <= 0:
82                 bullets.remove(bullet)
83
84         if collisions:
85             stats.score += ai_settings.enemy_points
86             sb.prep_score()
87             check_high_score(stats, sb)
88
89
90     def update_enemy(ai_settings, screen, stats, sb, player, enemies, bullets):
91         if pygame.sprite.spritecollideany(player, enemies):

```

check_enemies_l...

```

94
95 #to reset the player's position when collision with enemies
96 def ship_hit(ai_settings, screen,stats, sb,player, enemies, bullets):
97     if stats.ships_left > 0:
98         stats.ships_left -= 1
99         enemies.empty()
100         bullets.empty()
101         player.center_player()
102         sb.prep_player()
103
104         sleep(0.5)
105
106     else:
107         stats.game_active = False
108         pygame.mouse.set_visible(True)
109
110
111 def check_high_score(stats, sb):
112     """Check to see if there's a new highscore"""
113     if stats.score > stats.high_score:
114         stats.high_score = stats.score
115         sb.prep_high_score()
116
117 def check_enemies_left(ai_settings, screen, stats,sb, player,enemies, bullets):
118     """Check if enemies have reached the left screen"""
119     screen_rect = screen.get_rect()
120     for enemy in enemies.sprites():
121         if enemy.rect.left <= screen_rect.left:
122             ship_hit(ai_settings, screen,stats,sb, player, enemies, bullets)
123             break
124
125 check_enemies_l...

```

```

122         ship_hit(ai_settings, screen,stats,sb, player, enemies, bullets)
123         break
124
125 #function to update the screen
126 def update_screen(ai_settings, screen, stats, sb, player,enemies,bullets, play_button):
127     ai_settings.levelup(stats.score)
128
129     for bullet in bullets.sprites():
130         bullet.draw_bullet()
131     #to spawn random enemies
132     if pygame.time.get_ticks()%80 == 0:
133         new_enemy = Enemy(ai_settings, screen)
134         enemies.add(new_enemy)
135     if not stats.game_active:
136         play_button.draw_button()
137     sb.show_score()
138     player.blitme()
139
140     if stats.game_active:
141         for en in enemies:
142             en.update()
143             en.blitme()
144
145     pygame.display.flip()
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

check_enemies_l...

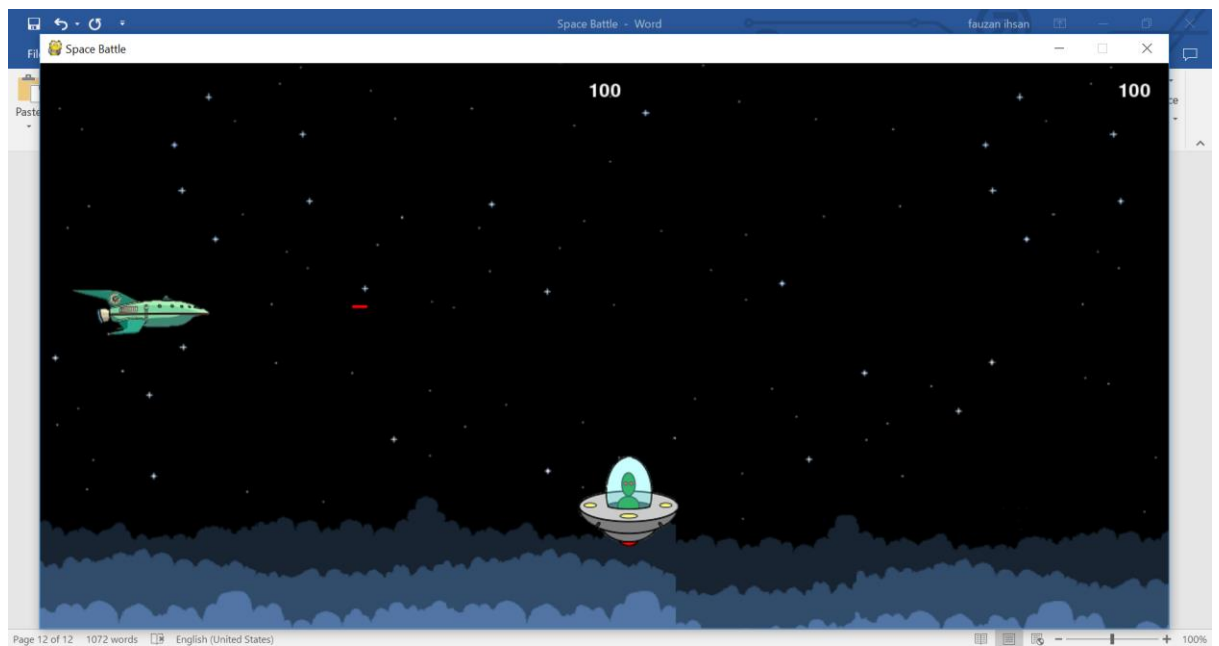
Main (Main.py) :

- *def main():*

This is the main function that runs and contains all of the other functions. Inside main function, there is the main loop that contains updates such as the *check_events*, *update_screen*, *update_bullets*, *update_enemy*, background and other event loop and codes that manages the screen.

```
1  import pygame
2  import sys
3
4
5  from settings import Settings
6  from player import Player
7  from pygame.sprite import Group
8  from enemy import Enemy
9  from game_stats import GameStats
10 from scoreboard import Scoreboard
11 from button import Button
12
13 import functions as gf
14
15 #the main function
16 def main():
17     pygame.init()
18     ai_settings = Settings()
19     screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_height))
20
21     bullets = Group()
22     enemies = Group()
23     enemybullets = Group()
24     enemy = Enemy(ai_settings, screen)
25
26     stats = GameStats(ai_settings)
27     sb = Scoreboard(ai_settings, screen, stats)
28
29     play_button = Button(ai_settings, screen, "Play")
30
31
32
33
34 player = Player(ai_settings, screen)
35 bground = pygame.image.load('images/wallhaven-41034.bmp')
36 x = 0
37
38
39 #main loop
40 while True:
41     gf.check_events(ai_settings, screen, stats,sb, play_button, player,enemies, bullets)
42
43     if stats.game_active:
44         player.update()
45         gf.update_bullets(ai_settings, screen, stats, sb, player, enemies,bullets)
46         gf.update_enemy(ai_settings, screen, stats,sb, player,enemies, bullets)
47
48     gf.update_screen(ai_settings, screen, stats, sb, player, enemies, bullets, play_button)
49
50     #moving background
51     screen.fill((0,0,0))
52     rel_x = x % bground.get_rect().width
53     screen.blit(bground, (rel_x - bground.get_rect().width,0))
54     if rel_x < ai_settings.screen_width:
55         screen.blit(bground, (rel_x,0))
56
57     x -= 3
58
59
60 main()
61
62 #Special thanks to longlong,arkaan,and python crash course for the help of making my final project#
63
```

IV. Evidence Of Working



V. References

1. Python Crash Course
2. www.youtube.com
3. www.stackoverflow.com