**PYGAME DAN KIVY**

Dimas Ramadhan | 180441100002

Dikky Jurais A | 180441100135

**Code Project Pygame**

Main.py

import pygame  
import sys  
import traceback  
from random import \*  
from pygame.locals import \*  
import bullet  
import myplane  
import enemy  
import supply  
import os  
  
  
pygame.init()  
pygame.mixer.init()  
bg\_size = width, height = 480, 720  
screen = pygame.display.set\_mode(bg\_size)  
pygame.display.set\_caption("GAME PBO SI")  
backgroud = pygame.image.load("image/background.png")  
  
  
pygame.mixer.music.load("sound/game\_music.mp3")  
pygame.mixer.music.set\_volume(0.2)  
bullet\_sound = pygame.mixer.Sound("sound/bullet.wav")  
bullet\_sound.set\_volume(0.2)  
big\_enemy\_flying\_sound = pygame.mixer.Sound("sound/big\_spaceship\_flying.wav")  
big\_enemy\_flying\_sound.set\_volume(0.2)  
enemy1\_down\_sound = pygame.mixer.Sound("sound/enemy1\_down.wav")  
enemy1\_down\_sound.set\_volume(0.2)  
enemy2\_down\_sound = pygame.mixer.Sound("sound/enemy2\_down.wav")  
enemy2\_down\_sound.set\_volume(0.2)  
enemy3\_down\_sound = pygame.mixer.Sound("sound/enemy3\_down.wav")  
enemy3\_down\_sound.set\_volume(0.2)  
me\_down\_sound = pygame.mixer.Sound("sound/game\_over.wav")  
me\_down\_sound.set\_volume(0.2)  
button\_down\_sound = pygame.mixer.Sound("sound/button.wav")  
button\_down\_sound.set\_volume(0.2)  
level\_up\_sound = pygame.mixer.Sound("sound/achievement.wav")  
level\_up\_sound.set\_volume(0.2)  
bomb\_sound = pygame.mixer.Sound("sound/use\_bomb.wav")  
bomb\_sound.set\_volume(0.2)  
get\_bomb\_sound = pygame.mixer.Sound("sound/get\_bomb.wav")  
get\_bomb\_sound.set\_volume(0.2)  
get\_bullet\_sound = pygame.mixer.Sound("sound/get\_double\_laser.wav")  
get\_bullet\_sound.set\_volume(0.2)  
  
  
  
def add\_small\_enemies(group1, group2, num):  
 for i in range(num):  
 e1 = enemy.SmallEnemy(bg\_size)  
 group1.add(e1)  
 group2.add(e1)  
  
  
def add\_mid\_enemies(group1, group2, num):  
 for i in range(num):  
 e2 = enemy.MidEnemy(bg\_size)  
 group1.add(e2)  
 group2.add(e2)  
  
  
def add\_big\_enemies(group1, group2, num):  
 for i in range(num):  
 e3 = enemy.BigEnemy(bg\_size)  
 group1.add(e3)  
 group2.add(e3)  
  
  
  
def inc\_speed(target, inc):  
 for each in target:  
 each.speed += inc  
  
  
def main():  
 clock = pygame.time.Clock()  
 switch\_image = False  
 delay = 60  
 pygame.mixer.music.play(-1)  
 running = True  
 me = myplane.MyPlane(bg\_size)  
 score = 0  
 score\_font = pygame.font.SysFont("arial", 48)  
  
 color\_black = (0, 0, 0)  
 color\_green = (0, 255, 0)  
 color\_red = (255, 0, 0)  
 color\_white = (255, 255, 255)  
  
 bomb\_image = pygame.image.load("image/bomb.png")  
 bomb\_rect = bomb\_image.get\_rect()  
 bomb\_font = score\_font  
 bomb\_num = 3  
  
 level = 1  
 paused = False  
 pause\_nor\_image = pygame.image.load("image/game\_pause\_nor.png")  
 pause\_pressed\_image = pygame.image.load("image/game\_pause\_pressed.png")  
 resume\_nor\_image = pygame.image.load("image/game\_resume\_nor.png")  
 resume\_pressed\_image = pygame.image.load("image/game\_resume\_pressed.png")  
 paused\_rect = pause\_nor\_image.get\_rect()  
 paused\_rect.left, paused\_rect.top = width - paused\_rect.width - 10, 10  
 paused\_image = pause\_nor\_image  
  
 gameover\_image = pygame.image.load("image/game\_over.png")  
 gameover\_rect = gameover\_image.get\_rect()  
 flag\_recorded = False  
  
 life\_image = pygame.image.load("image/life.png").convert()  
 life\_rect = life\_image.get\_rect()  
 life\_num = 3  
 invincible\_time = USEREVENT + 2  
  
 bullet1 = []  
 bullet1\_index = 0  
 bullet1\_num = 6  
 for i in range(bullet1\_num):  
 bullet1.append(bullet.Bullet1(me.rect.midtop))  
  
 double\_bullet\_timer = USEREVENT + 1  
 is\_double\_bullet = False  
 bullet2 = []  
 bullet2\_index = 0  
 bullet2\_num = 10  
 for i in range(bullet2\_num // 2):  
 bullet2.append(bullet.Bullet2((me.rect.centerx - 33, me.rect.centery)))  
 bullet2.append(bullet.Bullet2((me.rect.centerx + 30, me.rect.centery)))  
  
 enemies = pygame.sprite.Group()  
 small\_enemies = pygame.sprite.Group()  
 add\_small\_enemies(small\_enemies, enemies, 1)  
 mid\_enemies = pygame.sprite.Group()  
 add\_mid\_enemies(mid\_enemies, enemies, 1)  
 big\_enemies = pygame.sprite.Group()  
 add\_big\_enemies(big\_enemies, enemies, 1)  
  
 bullet\_supply = supply.BulletSupply(bg\_size)  
 bomb\_supply = supply.BombSupply(bg\_size)  
 supply\_timer = USEREVENT  
 pygame.time.set\_timer(supply\_timer, 10 \* 1000)  
  
 e1\_destroy\_index = 0  
 e2\_destroy\_index = 0  
 e3\_destroy\_index = 0  
 me\_destroy\_index = 0  
  
  
 while running:  
 screen.blit(backgroud, (0, 0))  
 score\_text = score\_font.render("Score: %s" % str(score), True, color\_white)  
 screen.blit(score\_text, (10, 5))  
  
 if level == 1 and score > 5000:  
 level = 2  
 level\_up\_sound.play()  
 add\_small\_enemies(small\_enemies, enemies, 3)  
 add\_mid\_enemies(mid\_enemies, enemies, 2)  
 add\_big\_enemies(big\_enemies, enemies, 1)  
 inc\_speed(small\_enemies, 1)  
 elif level == 2 and score > 30000:  
 level = 3  
 level\_up\_sound.play()  
 add\_small\_enemies(small\_enemies, enemies, 3)  
 add\_mid\_enemies(mid\_enemies, enemies, 2)  
 add\_big\_enemies(big\_enemies, enemies, 1)  
 inc\_speed(small\_enemies, 1)  
 inc\_speed(mid\_enemies, 1)  
 elif level == 3 and score > 60000:  
 level = 4  
 level\_up\_sound.play()  
 add\_small\_enemies(small\_enemies, enemies, 3)  
 add\_mid\_enemies(mid\_enemies, enemies, 2)  
 add\_big\_enemies(big\_enemies, enemies, 1)  
 inc\_speed(small\_enemies, 1)  
 inc\_speed(mid\_enemies, 1)  
 inc\_speed(big\_enemies, 1)  
  
  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == KEYDOWN:  
 if event.key == K\_SPACE:  
 if bomb\_num:  
 bomb\_num -= 1  
 bomb\_sound.play()  
 for each in enemies:  
 if each.rect.bottom > 0:  
 each.active = False  
 elif event.type == supply\_timer:  
 if choice([True, False]):  
 bomb\_supply.reset()  
 else:  
 bullet\_supply.reset()  
 elif event.type == double\_bullet\_timer:  
 is\_double\_bullet = False  
 pygame.time.set\_timer(double\_bullet\_timer, 0)  
 elif event.type == invincible\_time:  
 me.invincible = False  
 pygame.time.set\_timer(invincible\_time, 0)  
 elif event.type == MOUSEMOTION:  
 if paused\_rect.collidepoint(event.pos):  
 if paused:  
 paused\_image = resume\_pressed\_image  
 else:  
 paused\_image = pause\_pressed\_image  
 else:  
 if paused:  
 paused\_image = resume\_nor\_image  
 else:  
 paused\_image = pause\_nor\_image  
 elif event.type == MOUSEBUTTONDOWN:  
 button\_down\_sound.play()  
 if event.button == 1 and paused\_rect.collidepoint(event.pos):  
 paused = not paused  
 if paused:  
 paused\_image = resume\_pressed\_image  
 pygame.time.set\_timer(supply\_timer, 0)  
 pygame.mixer.music.pause()  
 pygame.mixer.pause()  
 else:  
 paused\_image = pause\_pressed\_image  
 pygame.time.set\_timer(supply\_timer, 30 \* 1000)  
 pygame.mixer.music.unpause()  
 pygame.mixer.unpause()  
 screen.blit(paused\_image, paused\_rect)  
  
 if life\_num and (not paused):  
  
 bomb\_text = bomb\_font.render("x %d" % bomb\_num, True, color\_black)  
 bomb\_text\_rect = bomb\_text.get\_rect()  
 screen.blit(bomb\_image, (10, height - 10 - bomb\_rect.height))  
 screen.blit(bomb\_text, (20 + bomb\_rect.width, height - 10 - bomb\_text\_rect.height))  
  
 for i in range(life\_num):  
 screen.blit(life\_image, (width - 10 - (i + 1) \* life\_rect.width, height - 10 - life\_rect.height))  
  
 key\_pressed = pygame.key.get\_pressed()  
 if key\_pressed[K\_w] or key\_pressed[K\_UP]:  
 me.move\_up()  
 if key\_pressed[K\_s] or key\_pressed[K\_DOWN]:  
 me.move\_down()  
 if key\_pressed[K\_a] or key\_pressed[K\_LEFT]:  
 me.move\_left()  
 if key\_pressed[K\_d] or key\_pressed[K\_RIGHT]:  
 me.move\_right()  
  
 if not (delay % 10):  
 bullet\_sound.play()  
 if not is\_double\_bullet:  
 bullets = bullet1  
 bullets[bullet1\_index].reset(me.rect.midtop)  
 bullet1\_index = (bullet1\_index + 1) % bullet1\_num  
 else:  
 bullets = bullet2  
 bullets[bullet2\_index].reset((me.rect.centerx - 33, me.rect.centery))  
 bullets[bullet2\_index + 1].reset((me.rect.centerx + 30, me.rect.centery))  
 bullet2\_index = (bullet2\_index + 2) % bullet2\_num  
  
 if bomb\_supply.active:  
 bomb\_supply.move()  
 screen.blit(bomb\_supply.image, bomb\_supply.rect)  
 if pygame.sprite.collide\_mask(bomb\_supply, me):  
 get\_bomb\_sound.play()  
 if bomb\_num < 3:  
 bomb\_num += 1  
 bomb\_supply.active = False  
 if bullet\_supply.active:  
 bullet\_supply.move()  
 screen.blit(bullet\_supply.image, bullet\_supply.rect)  
 if pygame.sprite.collide\_mask(bullet\_supply, me):  
 get\_bullet\_sound.play()  
 is\_double\_bullet = True  
 pygame.time.set\_timer(double\_bullet\_timer, 18 \* 1000)  
 bullet\_supply.active = False  
  
 for b in bullets:  
 if b.active:  
 b.move()  
 screen.blit(b.image, b.rect)  
 enemies\_hit = pygame.sprite.spritecollide(b, enemies, False, pygame.sprite.collide\_mask)  
 if enemies\_hit:  
 b.active = False  
 for e in enemies\_hit:  
 if e in big\_enemies or e in mid\_enemies:  
 e.energy -= 1  
 e.hit = True  
 if e.energy == 0:  
 e.active = False  
 else:  
 e.active = False  
  
 enemies\_down = pygame.sprite.spritecollide(me, enemies, False, pygame.sprite.collide\_mask)  
 if enemies\_down and not me.invincible:  
 me.active = False  
 for e in enemies\_down:  
 e.active = False  
  
  
 if delay == 0:  
 delay = 60  
 delay -= 1  
 if not delay % 3:  
 switch\_image = not switch\_image  
 if me.active:  
 if switch\_image:  
 screen.blit(me.image1, me.rect)  
 else:  
 screen.blit(me.image2, me.rect)  
 else:  
 if not (delay % 3):  
 screen.blit(me.destroy\_images[me\_destroy\_index], me.rect)  
 me\_destroy\_index = (me\_destroy\_index + 1) % 4  
 if me\_destroy\_index == 0:  
 me\_down\_sound.play()  
 life\_num -= 1  
 me.reset()  
 pygame.time.set\_timer(invincible\_time, 3 \* 1000)  
  
 for each in big\_enemies:  
 if each.active:  
 each.move()  
 if not each.hit:  
 if switch\_image:  
 screen.blit(each.image1, each.rect)  
 else:  
 screen.blit(each.image2, each.rect)  
 else:  
 screen.blit(each.image\_hit, each.rect)  
 each.hit = False  
  
 pygame.draw.line(screen, color\_black, (each.rect.left, each.rect.top - 5),  
 (each.rect.right, each.rect.top - 5), 2)  
 energy\_remain = each.energy / enemy.BigEnemy.energy  
 if energy\_remain > 0.2:  
 energy\_color = color\_green  
 else:  
 energy\_color = color\_red  
 pygame.draw.line(screen, energy\_color, (each.rect.left, each.rect.top - 5),  
 (each.rect.left + each.rect.width \* energy\_remain, each.rect.top - 5), 2)  
 if each.rect.bottom == -50:  
 big\_enemy\_flying\_sound.play(-1)  
 else:  
 big\_enemy\_flying\_sound.stop()  
 if e3\_destroy\_index == 0:  
 enemy3\_down\_sound.play()  
 if not (delay % 3):  
 screen.blit(each.destroy\_images[e3\_destroy\_index], each.rect)  
 e3\_destroy\_index = (e3\_destroy\_index + 1) % 6  
 if e3\_destroy\_index == 0:  
 score += 6000  
 each.reset()  
  
 for each in mid\_enemies:  
 if each.active:  
 each.move()  
 if not each.hit:  
 screen.blit(each.image, each.rect)  
 else:  
 screen.blit(each.image\_hit, each.rect)  
 each.hit = False  
  
 pygame.draw.line(screen, color\_black,  
 (each.rect.left, each.rect.top - 5),  
 (each.rect.right, each.rect.top - 5),  
 2)  
 energy\_remain = each.energy / enemy.MidEnemy.energy  
 if energy\_remain > 0.2:  
 energy\_color = color\_green  
 else:  
 energy\_color = color\_red  
 pygame.draw.line(screen, energy\_color,  
 (each.rect.left, each.rect.top - 5),  
 (each.rect.left + each.rect.width \* energy\_remain, each.rect.top - 5),  
 2)  
 else:  
 if e2\_destroy\_index == 0:  
 enemy2\_down\_sound.play()  
 if not (delay % 3):  
 screen.blit(each.destroy\_images[e2\_destroy\_index], each.rect)  
 e2\_destroy\_index = (e2\_destroy\_index + 1) % 4  
 if e2\_destroy\_index == 0:  
 score += 2000  
 each.reset()  
  
 for each in small\_enemies:  
 if each.active:  
 each.move()  
 screen.blit(each.image, each.rect)  
 else:  
 if e1\_destroy\_index == 0:  
 enemy1\_down\_sound.play()  
 if not (delay % 3):  
 screen.blit(each.destroy\_images[e1\_destroy\_index], each.rect)  
 e1\_destroy\_index = (e1\_destroy\_index + 1) % 4  
 if e1\_destroy\_index == 0:  
 score += 500  
 each.reset()  
  
 elif life\_num == 0:  
 screen.blit(gameover\_image, gameover\_rect)  
 pygame.mixer.music.stop()  
 pygame.mixer.stop()  
 pygame.time.set\_timer(supply\_timer, 0)  
  
 if not flag\_recorded:  
 flag\_recorded = True  
 if not os.path.exists("score\_record.txt"):  
 with open("score\_record.txt", "w") as f:  
 f.write('0')  
 with open("score\_record.txt", "r") as f:  
 record\_score = int(f.read())  
 if score > record\_score:  
 with open("score\_record.txt", "w") as f:  
 f.write(str(score))  
 f.close()  
  
 record\_score\_text = score\_font.render("%d" % record\_score, True, color\_black)  
 screen.blit(record\_score\_text, (50, 25))  
 game\_over\_score\_text = score\_font.render("%d" % score, True, color\_black)  
 screen.blit(game\_over\_score\_text, (300, 25))  
  
 pygame.display.flip()  
 clock.tick(60)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 try:  
 main()  
 except SystemExit:  
 pass  
 except:  
 traceback.print\_exc()  
 pygame.quit()  
 input()

bullet.py

import pygame  
  
  
class Bullet1(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, position):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.image.load("image/bullet1.png")  
 self.rect = self.image.get\_rect()  
 self.rect.left, self.rect.top = position  
 self.speed = 12  
 self.active = True  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 def move(self):  
 if self.rect.top < 0:  
 self.active = False  
 else:  
 self.rect.top -= self.speed  
  
 def reset(self, position):  
 self.rect.left, self.rect.top = position  
 self.active = True  
  
  
class Bullet2(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, position):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.image.load("image/bullet2.png")  
 self.rect = self.image.get\_rect()  
 self.rect.left, self.rect.top = position  
 self.speed = 14  
 self.active = True  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 def move(self):  
 if self.rect.top < 0:  
 self.active = False  
 else:  
 self.rect.top -= self.speed  
  
 def reset(self, position):  
 self.rect.left, self.rect.top = position  
 self.active = True

enemy.py

import pygame  
from random import \*  
class SmallEnemy(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.image.load("image/enemy1.png")  
 self.mask = pygame.mask.from\_surface(self.image)  
 self.destroy\_images = []  
 self.destroy\_images.extend([pygame.image.load("image/enemy1\_down1.png"),  
 pygame.image.load("image/enemy1\_down2.png"),  
 pygame.image.load("image/enemy1\_down3.png"),  
 pygame.image.load("image/enemy1\_down4.png")])  
 self.rect = self.image.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.speed = 2  
 self.rect.left, self.rect.top = (randint(0, self.width - self.rect.width),  
 randint(-5 \* self.rect.height, -5)  
 )  
 self.active = True  
  
 def move(self):  
 if self.rect.top < self.height:  
 self.rect.top += self.speed  
 else:  
 self.reset()  
  
 def reset(self):  
 self.rect.left, self.rect.top = (randint(0, self.width - self.rect.width),  
 randint(-5 \* self.rect.height, 0)  
 )  
 self.active = True  
  
  
class MidEnemy(pygame.sprite.Sprite):  
 energy = 5  
  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.image.load("image/enemy2.png")  
 self.mask = pygame.mask.from\_surface(self.image)  
 self.destroy\_images = []  
 self.destroy\_images.extend([pygame.image.load("image/enemy2\_down1.png"),  
 pygame.image.load("image/enemy2\_down2.png"),  
 pygame.image.load("image/enemy2\_down3.png"),  
 pygame.image.load("image/enemy2\_down4.png")])  
 self.image\_hit = pygame.image.load("image/enemy2\_hit.png")  
 self.rect = self.image.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.speed = 1  
 self.rect.left, self.rect.top = (randint(0, self.width - self.rect.width),  
 randint(-10 \* self.rect.height, -self.rect.height)  
 )  
 self.active = True  
 self.energy = MidEnemy.energy  
 self.hit = False  
  
 def move(self):  
 if self.rect.top < self.height:  
 self.rect.top += self.speed  
 else:  
 self.reset()  
  
 def reset(self):  
 self.rect.left, self.rect.top = (randint(0, self.width - self.rect.width),  
 randint(-10 \* self.rect.height, -self.rect.height)  
 )  
 self.active = True  
 self.energy = MidEnemy.energy  
 self.hit = False  
  
  
class BigEnemy(pygame.sprite.Sprite):  
 energy = 15  
  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image1 = pygame.image.load("image/enemy3\_n1.png")  
 self.image2 = pygame.image.load("image/enemy3\_n2.png")  
 self.mask = pygame.mask.from\_surface(self.image1)  
 self.destroy\_images = []  
 self.destroy\_images.extend([pygame.image.load("image/enemy3\_down1.png"),  
 pygame.image.load("image/enemy3\_down2.png"),  
 pygame.image.load("image/enemy3\_down3.png"),  
 pygame.image.load("image/enemy3\_down4.png"),  
 pygame.image.load("image/enemy3\_down5.png"),  
 pygame.image.load("image/enemy3\_down6.png")])  
 self.image\_hit = pygame.image.load("image/enemy3\_hit.png")  
 self.rect = self.image1.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.speed = 2  
 self.rect.left, self.rect.top = (randint(0, self.width - self.rect.width),  
 randint(-15 \* self.rect.height, -5 \* self.rect.height)  
 )  
 self.active = True  
 self.energy = BigEnemy.energy  
 self.hit = False  
  
 def move(self):  
 if self.rect.top < self.height:  
 self.rect.top += self.speed  
 else:  
 self.reset()  
  
 def reset(self):  
 self.rect.left, self.rect.top = (randint(0, self.width - self.rect.width),  
 randint(-15 \* self.rect.height, -5 \* self.rect.height)  
 )  
 self.active = True  
 self.energy = BigEnemy.energy  
 self.hit = False

myplane.py

import pygame  
  
  
class MyPlane(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image1 = pygame.image.load("image/hero1.png")  
 self.image2 = pygame.image.load("image/hero2.png")  
 self.mask = pygame.mask.from\_surface(self.image1)  
 self.destroy\_images = []  
 self.destroy\_images.extend([pygame.image.load("image/hero\_blowup\_n1.png"),  
 pygame.image.load("image/hero\_blowup\_n2.png"),  
 pygame.image.load("image/hero\_blowup\_n3.png"),  
 pygame.image.load("image/hero\_blowup\_n4.png")])  
 self.rect = self.image1.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.rect.left, self.rect.top = (self.width - self.rect.width) // 2, (  
 self.height - self.rect.height - 60)  
 self.speed = 10  
 self.active = True  
 self.invincible = False  
  
  
 def move\_up(self):  
 if self.rect.top > 0:  
 self.rect.top -= self.speed  
 else:  
 self.rect.top = 0  
  
 def move\_down(self):  
 if self.rect.bottom < self.height - 60:  
 self.rect.top += self.speed  
 else:  
 self.rect.bottom = self.height - 60  
  
 def move\_left(self):  
 if self.rect.left > 0:  
 self.rect.left -= self.speed  
 else:  
 self.rect.left = 0  
  
 def move\_right(self):  
 if self.rect.right < self.width:  
 self.rect.right += self.speed  
 else:  
 self.rect.right = self.width  
  
 def reset(self):  
 self.rect.left, self.rect.top = (self.width - self.rect.width) // 2, (self.height - self.rect.height - 60)  
 self.active = True  
 self.invincible = True

supply.py

import pygame  
from random import \*  
class BombSupply(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("image/ufo2.png")  
 self.rect = self.image.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.rect.left, self.rect.bottom = randint(0, self.width - self.rect.width), -100  
 self.speed = 5  
 self.active = False  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 def move(self):  
 if self.rect.top < self.height:  
 self.rect.top += self.speed  
 else:  
 self.active = False  
  
 def reset(self):  
 self.active = True  
 self.rect.left, self.rect.bottom = randint(0, self.width - self.rect.width), -100  
  
  
  
class BulletSupply(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("image/ufo1.png")  
 self.rect = self.image.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.rect.left, self.rect.bottom = randint(0, self.width - self.rect.width), -100  
 self.speed = 5  
 self.active = False  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 def move(self):  
 if self.rect.top < self.height:  
 self.rect.top += self.speed  
 else:  
 self.active = False  
  
 def reset(self):  
 self.active = True  
 self.rect.left, self.rect.bottom = randint(0, self.width - self.rect.width), -100

**Code Project Kivy**

Main.py

from kivy.app import App  
from kivy.lang import Builder  
from kivy.uix.screenmanager import ScreenManager, Screen  
from kivy.properties import ObjectProperty  
from kivy.uix.popup import Popup  
from kivy.uix.label import Label  
from database import DataBase  
  
  
class CreateAccountWindow(Screen):  
 namee = ObjectProperty(None)  
 email = ObjectProperty(None)  
 password = ObjectProperty(None)  
  
 def submit(self):  
 if self.namee.text != "" and self.email.text != "" and self.email.text.count("@") == 1 and self.email.text.count(".") > 0:  
 if self.password != "":  
 db.add\_user(self.email.text, self.password.text, self.namee.text)  
  
 self.reset()  
  
 sm.current = "login"  
 else:  
 invalidForm()  
 else:  
 invalidForm()  
  
 def login(self):  
 self.reset()  
 sm.current = "login"  
  
 def reset(self):  
 self.email.text = ""  
 self.password.text = ""  
 self.namee.text = ""  
  
  
class LoginWindow(Screen):  
 email = ObjectProperty(None)  
 password = ObjectProperty(None)  
  
 def loginBtn(self):  
 if db.validate(self.email.text, self.password.text):  
 MainWindow.current = self.email.text  
 self.reset()  
 sm.current = "main"  
 else:  
 invalidLogin()  
  
 def createBtn(self):  
 self.reset()  
 sm.current = "create"  
  
 def reset(self):  
 self.email.text = ""  
 self.password.text = ""  
  
  
class MainWindow(Screen):  
 n = ObjectProperty(None)  
 created = ObjectProperty(None)  
 email = ObjectProperty(None)  
 current = ""  
  
 def logOut(self):  
 sm.current = "login"  
  
 def on\_enter(self, \*args):  
 password, name, created = db.get\_user(self.current)  
 self.n.text = "Account Name: " + name  
 self.email.text = "Email: " + self.current  
 self.created.text = "Created On: " + created  
  
  
class WindowManager(ScreenManager):  
 pass  
  
  
def invalidLogin():  
 pop = Popup(title='Invalid Login',  
 content=Label(text='Invalid username or password.'),  
 size\_hint=(None, None), size=(400, 400))  
 pop.open()  
  
  
def invalidForm():  
 pop = Popup(title='Invalid Form',  
 content=Label(text='Please fill in all inputs with valid information.'),  
 size\_hint=(None, None), size=(400, 400))  
  
 pop.open()  
  
  
kv = Builder.load\_file("my.kv")  
  
sm = WindowManager()  
db = DataBase("users.txt")  
  
screens = [LoginWindow(name="login"), CreateAccountWindow(name="create"),MainWindow(name="main")]  
for screen in screens:  
 sm.add\_widget(screen)  
  
sm.current = "login"  
  
  
class MyMainApp(App):  
 def build(self):  
 return sm  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 MyMainApp().run()

my.kv

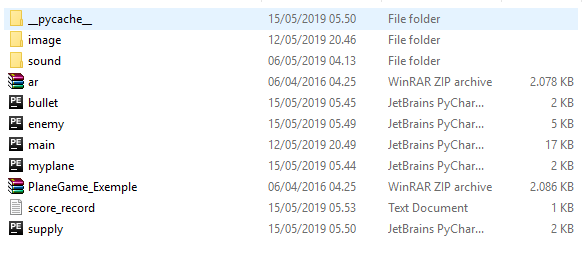
<CreateAccountWindow>:  
 name: "create"  
  
 namee: namee  
 email: email  
 password: passw  
  
 FloatLayout:  
 cols:1  
  
 FloatLayout:  
 size: root.width, root.height/2  
  
 Label:  
 text: "BUAT AKUN UTM ANDA"  
 size\_hint: 0.8, 0.2  
 pos\_hint: {"x":0.1, "top":1}  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 Label:  
 size\_hint: 0.5,0.12  
 pos\_hint: {"x":0, "top":0.8}  
 text: "Name: "  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 TextInput:  
 pos\_hint: {"x":0.5, "top":0.8}  
 size\_hint: 0.4, 0.12  
 id: namee  
 multiline: False  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 Label:  
 size\_hint: 0.5,0.12  
 pos\_hint: {"x":0, "top":0.8-0.13}  
 text: "Email: "  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 TextInput:  
 pos\_hint: {"x":0.5, "top":0.8-0.13}  
 size\_hint: 0.4, 0.12  
 id: email  
 multiline: False  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 Label:  
 size\_hint: 0.5,0.12  
 pos\_hint: {"x":0, "top":0.8-0.13\*2}  
 text: "Password: "  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 TextInput:  
 pos\_hint: {"x":0.5, "top":0.8-0.13\*2}  
 size\_hint: 0.4, 0.12  
 id: passw  
 multiline: False  
 password: True  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
  
 Button:  
 pos\_hint:{"x":0.3,"y":0.25}  
 size\_hint: 0.4, 0.1  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 17\*\*4  
 text: "Sudah Punya Akun UTM? Masuk"  
 on\_release:  
 root.manager.transition.direction = "left"  
 root.login()  
  
 Button:  
 pos\_hint:{"x":0.2,"y":0.05}  
 size\_hint: 0.6, 0.15  
 text: "Selesai"  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 14\*\*4  
 on\_release:  
 root.manager.transition.direction = "left"  
 root.submit()  
  
  
<LoginWindow>:  
 name: "login"  
  
 email: email  
 password: password  
  
 FloatLayout:  
  
 Label:  
 text:"Email: "  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 13\*\*4  
 pos\_hint: {"x":0.1, "top":0.9}  
 size\_hint: 0.35, 0.15  
  
 TextInput:  
 id: email  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 13\*\*4  
 multiline: False  
 pos\_hint: {"x": 0.45 , "top":0.9}  
 size\_hint: 0.4, 0.15  
  
 Label:  
 text:"Password: "  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 13\*\*4  
 pos\_hint: {"x":0.1, "top":0.7}  
 size\_hint: 0.35, 0.15  
  
 TextInput:  
 id: password  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 13\*\*4  
 multiline: False  
 password: True  
 pos\_hint: {"x": 0.45, "top":0.7}  
 size\_hint: 0.4, 0.15  
  
 Button:  
 pos\_hint:{"x":0.2,"y":0.05}  
 size\_hint: 0.6, 0.2  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 13\*\*4  
 text: "Login"  
 on\_release:  
 root.manager.transition.direction = "up"  
 root.loginBtn()  
  
 Button:  
 pos\_hint:{"x":0.3,"y":0.3}  
 size\_hint: 0.4, 0.1  
 font\_size: (root.width\*\*2 + root.height\*\*2) / 17\*\*4  
 text: "Belum Punya Akun UTM? Ayo Buat"  
 on\_release:  
 root.manager.transition.direction = "right"  
 root.createBtn()  
  
  
<MainWindow>:  
 n: n  
 email: email  
 created:created  
  
 FloatLayout:  
 Label:  
 id: n  
 pos\_hint:{"x": 0.1, "top":0.9}  
 size\_hint:0.8, 0.2  
 text: "Nama Akun: "  
  
 Label:  
 id: email  
 pos\_hint:{"x": 0.1, "top":0.7}  
 size\_hint:0.8, 0.2  
 text: "Email: "  
  
 Label:  
 id: created  
 pos\_hint:{"x": 0.1, "top":0.5}  
 size\_hint:0.8, 0.2  
 text: "Created: "  
  
 Button:  
 pos\_hint:{"x":0.2, "y": 0.1}  
 size\_hint:0.6,0.2  
 text: "Log Out"  
 on\_release:  
 app.root.current = "login"  
 root.manager.transition.direction = "down"

Database.py

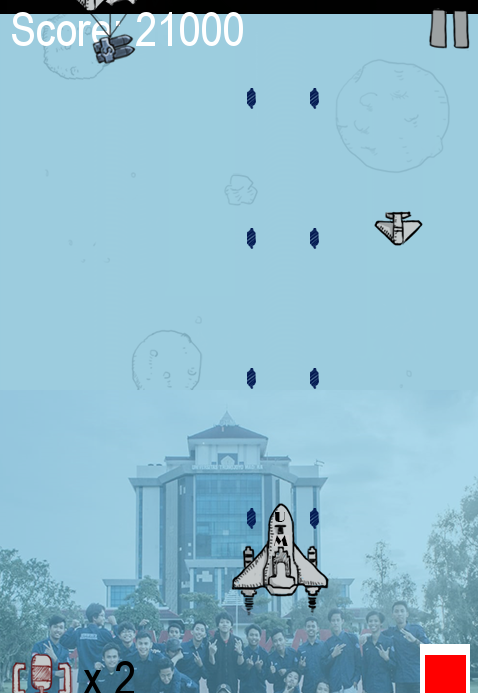
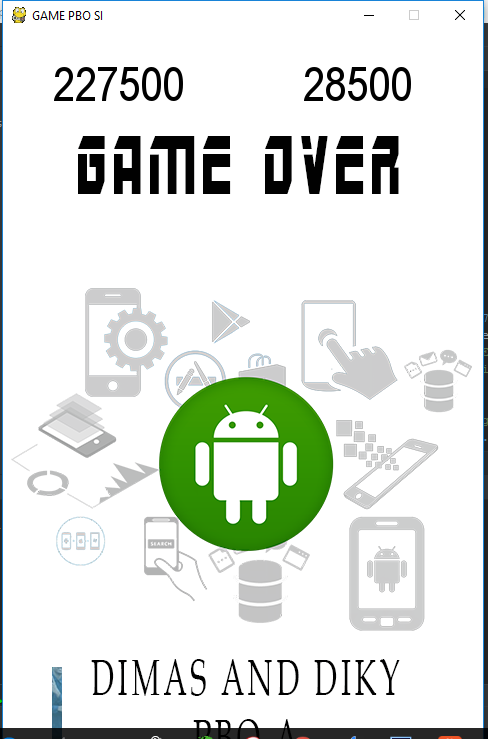
import datetime  
  
  
class DataBase:  
 def \_\_init\_\_(self, filename):  
 self.filename = filename  
 self.users = None  
 self.file = None  
 self.load()  
  
 def load(self):  
 self.file = open(self.filename, "r")  
 self.users = {}  
  
 for line in self.file:  
 email, password, name, created = line.strip().split(";")  
 self.users[email] = (password, name, created)  
  
 self.file.close()  
  
 def get\_user(self, email):  
 if email in self.users:  
 return self.users[email]  
 else:  
 return -1  
  
 def add\_user(self, email, password, name):  
 if email.strip() not in self.users:  
 self.users[email.strip()] = (password.strip(), name.strip(), DataBase.get\_date())  
 self.save()  
 return 1  
 else:  
 print("Email exists already")  
 return -1  
  
 def validate(self, email, password):  
 if self.get\_user(email) != -1:  
 return self.users[email][0] == password  
 else:  
 return False  
  
 def save(self):  
 with open(self.filename, "w") as f:  
 for user in self.users:  
 f.write(user + ";" + self.users[user][0] + ";" + self.users[user][1] + ";" + self.users[user][2] + "\n")  
  
 @staticmethod  
 def get\_date():  
 return str(datetime.datetime.now()).split(" ")[0]

**HASIL Screen Shoot Program Pygame**

**File Folder Pygame Aircraft War**

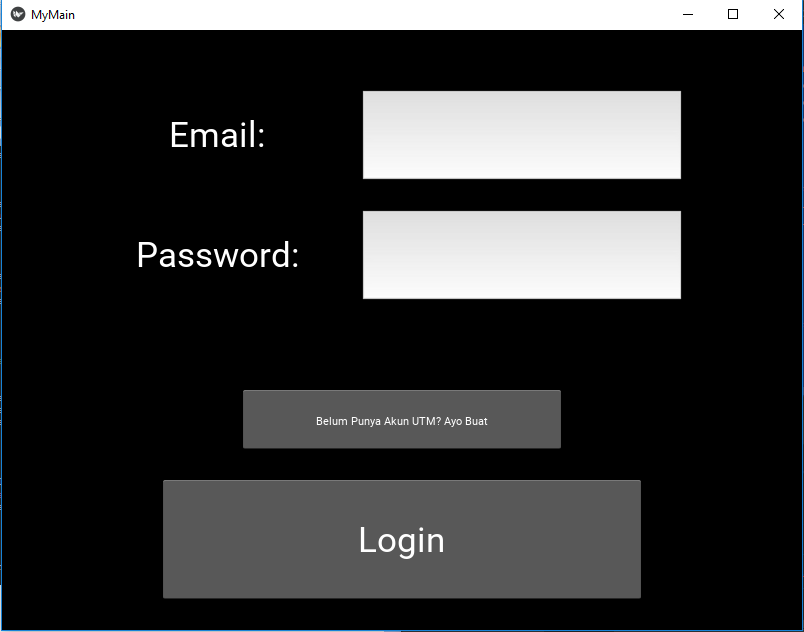


Gameplay

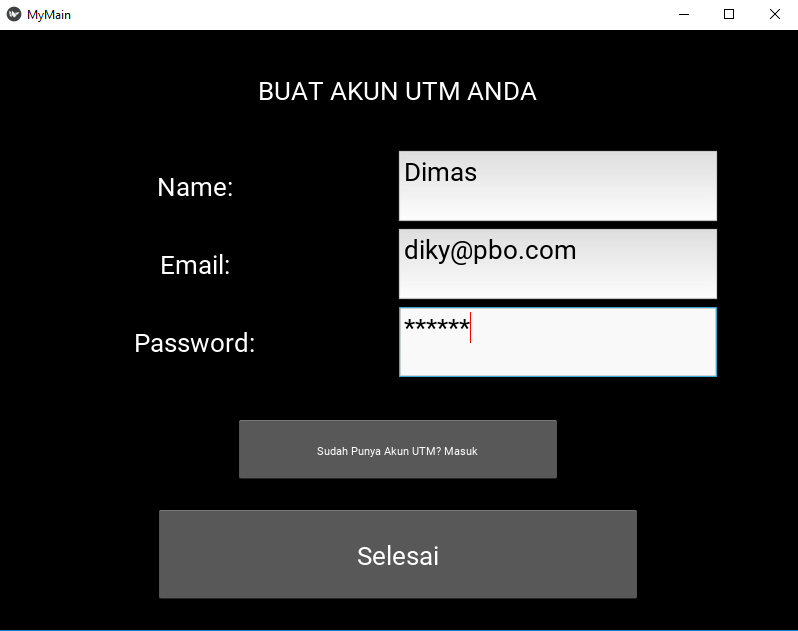


**HASIL Screen Shoot Program Kivy**

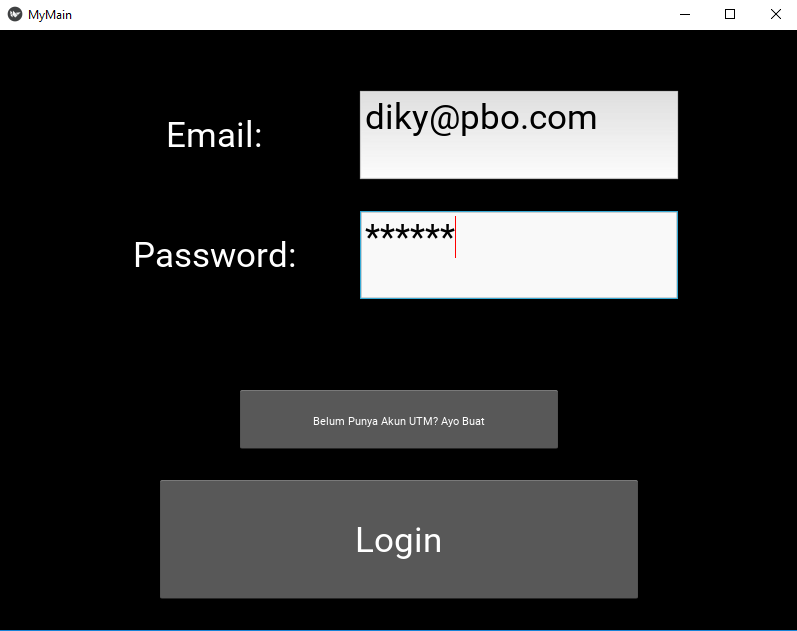
**File Folder Kivy Form Login**



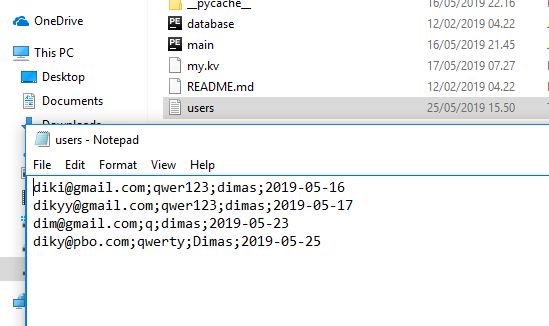
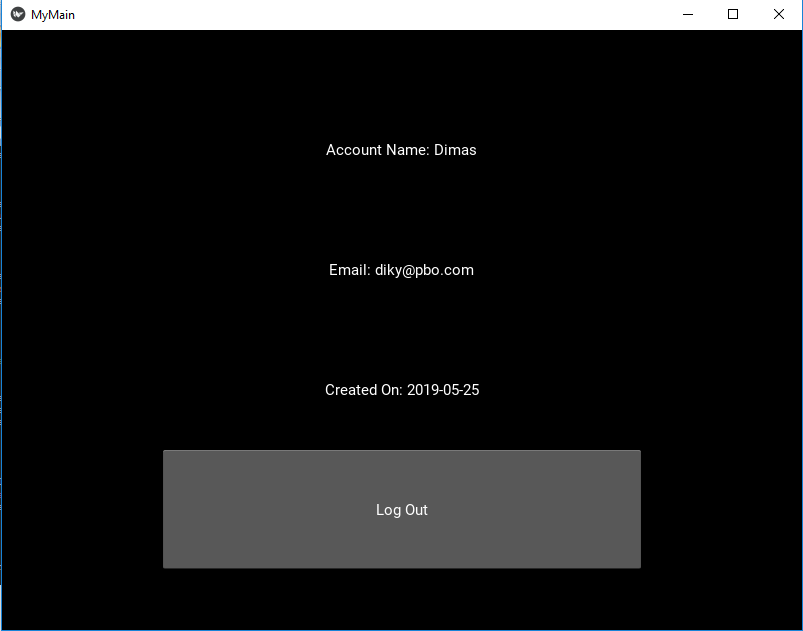
Buat Akun



Login



Hasil



**Pengertian Dan Cara Instalasi**

**PYGAME**

**Pengertian**

Pygame adalah seperangkat modul Python yang dirancang untuk membuat permainan. Pygame menambahkan fungsi di atas dengan sangat baik di SDL perpustakaan. Hal ini memungkinkan Anda untuk membuat sebuah game dengan fitur yang lengkap dan sebuah program multimedia dalam bahasa python. Pygame sangat portabel dan dapat berjalan pada hampir semua platform dan sistem operasi. Pygame sendiri telah didownload jutaan kali, dan telah memiliki jutaan kunjungan ke situsnya.

Pygame ini gratis. Dan dirilis di bawah Lisensi GPL , Anda dapat membuat sebuah aplikasi yang open source, gratis, freeware, shareware, dan game komersial dengan pygame ini.Untuk pengenalan yang bagus untuk pygame ini, periksa baris demi baris simpanse tutorial , dan pengenalan untuk programmer python atau Bab 17 sampai 20 dari buku "Ciptakan Anda Game Komputer Sendiri dengan Python" .

**Cara instalasi**

1. Cek versi python kalian

kalian harus mengetahui terlebih dahulu versi python yang terinstall di PC kalian dulu ya….

### Medownload atau mengunduh file instalasi pygame.

### Setelah itu kalian buka browser kalian dan pergi ke [pygame.org/download.shtml](http://pygame.org/download.shtml) . nah terus, klik link seperti Pada gambar bawah ini. Atau lebih mudahnya, kalian bisa klik link ini [http://www.lfd.uci.edu~gohlke/pythonlibs/#pygame](#pygame)

### extract file hasil download.

### Nah setelah itu, silahkan cari file instalasi pygame hasil download nya tadi ya… setelah itu, ubah extensi “.whl” menjadi “.zip”

### Paste file yang sudah di copy.

### Nah setelah kalian salin file – file nya, Lalu selanjutnya silahkan kalian pergi ke drive “C”. lalu pilih folder python sesuai versi nya. Contoh “pythonXX” huruf XX ini menandakan versi python nya. Contoh “python34”. Silahkan masuk ke folder “pythonxx” nya.. lalu masuk ke folder “include”. Nah di folder ini silahkan buat folder baru dengan nama “pygame”. Lalu paste file yang sudah di copy tadi kedalam folder “pygame”

### copy paste folder pygame.

### Setelah itu selesai, silahkan copy folder “pygame” dan “pygame-1.9.2.data” hasil extract tadi ya. INGATT!!! Folder ”pygame” yang di copy adalah folder hasil extract nya ya.. bukan folder yang kita buat tadi.

### Finsih dan pengecekan.

Nah sekarang sudah selesai proses nya… namun, untuk memastikan pygame nya sudah terinstall atau belum, sekarang buka IDLE nya.. dan coba ketikkan “import pygame”

**KIVY**

**Pengertian**

Kivy merupakan library Python open source yang digunakan untuk mengembangkan aplikasi mobile dengan user inetrface alami (Natural User Interface). Kivy bisa berjalan pada platform Android, iOS, Linux, OS X dan Windows. Kivy berlisensi MIT. (Wikipedia).

**Cara Instalasi**

1. Pastikan Anda memiliki pip dan roda terbaru

*python -m pip install --upgrade pip wheel setuptools*

1. Instal dependensi (lewati gstreamer (~ 120MB) jika tidak diperlukan, lihat dependensi Kivy):

*python -m pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew*

*python -m pip install kivy.deps.gstreamer*

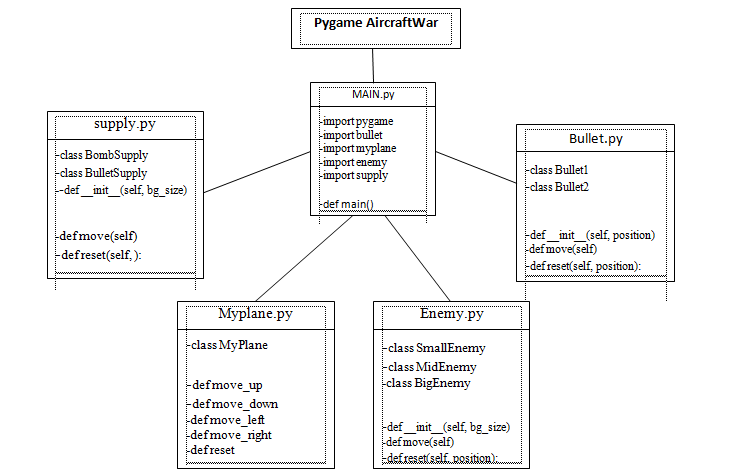
1. Untuk Python 3.5+, Anda juga bisa menggunakan backend angle bukan glew. Ini dapat diinstal dengan:

*python -m pip install kivy.deps.angle*

*python -m pip install kivy*

**Struktur Kelas Dan Penjelasan**

Di sini Saya Menggunkaan Berbasis File (.py) jadi setuap Class atau Orientasi nya terpisah di setiap file dan . Kemudian Akan Di Gabungkan dengan Cara Di Import kan ke file Utama (Main.py**)**



**Global Variable:**

import pygame (memasukan Pygame)  
import sys  
import traceback  
from random import \*  
from pygame.locals import \*  
import bullet (Memasukan Class atau file Bullet / peluru)  
import myplane (Memasukan Class atau file plane /pesawat)  
import enemy (Memasukan Class atau file enemy /musuh)  
import supply (Memasukan Class atau file supply /logistik)  
import os

Pygame.init()

bg\_size = width, height (Ukuran gameplay)

screen = width, height (Ukuran layar Screen pygame )

**Method :**

Init

Add

sprite.Sprite

image

rect

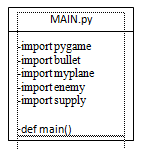
display.set

surface.rect

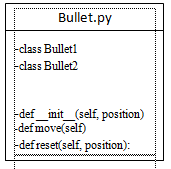
mask.from\_surface

images.extend

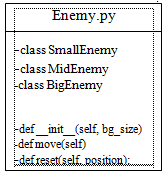
**Main.py**



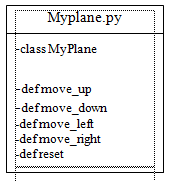
Berbasis File (.py) jadi setuap Class atau Orientasi nya terpisah di setiap file dan . Kemudian Akan Di Gabungkan dengan Cara Di Import kan ke file Utama (Main.py**)**



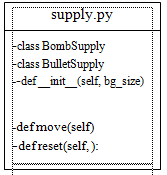
* Class bullet 1 dan 2 (merukapan peluru awal dan peluru ke dua setelah mendapat suply)
* def.move(self) (untuk gerakan dari peluru tersebut speed dan arah nya)
* def.reset (untuk mereset pergerakan peluru)



* calss smallEnemy (adalah musuh kecil yang biasa 1 hit kalah)
* calss midlEnemy (adalah musuh sedang yang 3 hit kalah)
* calss biglEnemy (adalah musuh besar raja yang 6 hit kalah)
* def.move(self) (untuk gerakan dari lawan tersebut speed dan arah nya)
* def.reset (untuk mereset pergerakan lawam)

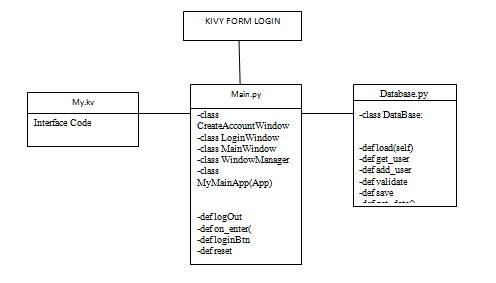


* class Myplane (adalah class dari pesawat kita)
* def.move(self/ all) (untuk gerakan dari pesawat kita tersebut speed dan arah nya)
* def.reset (untuk mereset pergerakan peawat kita)



* class boom suply (suplay boom atau senjata)
* class bullet (adalah peluru 2 pada bullet)
* def.move(self) (untuk gerakan dari objek tersebut speed dan arah nya)
* def.reset (untuk mereset pergerakan objek)

**Kivy**



Hampir sama dengan pygame kivy ada file yang berbentuk (.kv) yang merukapan desain atau interface pada aplikasi yang ber dasar pada kode python

**IMPLEMENTASI MATERI OOP**

**Property** (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah **class**. Melanjutkan analogi tentang laptop, **property** dari laptop bisa berupa merk, warna, jenis processor, ukuran layar, dan lain-lain.

class SmallEnemy(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.image.load("image/enemy1.png")  
 self.mask = pygame.mask.from\_surface(self.image)  
 self.destroy\_images = []  
 self.destroy\_images.extend

**Objek** adalah hasil cetak dari **class**, atau hasil ‘konkrit’ dari **class**. Jika menggunakan analogi **class laptop**, maka objek dari **class laptop** bisa berupa: laptop\_andi, laptop\_anto, laptop\_duniailkom, dan lain-lain. Objek dari **class laptop**akan memiliki seluruh ciri-ciri **laptop**, yaitu property dan method-nya.

pygame.display.flip()

**Method** adalah tindakan yang bisa dilakukan di dalam class. Jika menggunakan analogi **class laptop**kita, maka contoh method adalah: **menghidupkan laptop**, **mematikan laptop**, **mengganti cover laptop**, dan berbagai tindakan lain.

class BulletSupply(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("image/ufo1.png")  
 self.rect = self.image.get\_rect()  
 self.width, self.height = bg\_size[0], bg\_size[1]  
 self.rect.left, self.rect.bottom = randint(0, self.width - self.rect.width), -100  
 self.speed = 5  
 self.active = False  
 self.mask = pygame.mask.from\_surface(self.image)

Class adalah ‘cetak biru’ atau ‘blueprint’ dari **object**. Class digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nantinya adalah hasil cetakan dari class, yakni **object**.

import pygame  
  
  
class MyPlane(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image1 = pygame.image.load("image/hero1.png")  
 self.image2 = pygame.image.load("image/hero2.png")  
 self.mask = pygame.mask.from\_surface(self.image1)

**Inheritance** (warisan atau turunan) adalah sebuah konsep yang penting di dalam**Python**. **Inheritance** adalah sebuah proses dimana sebuah class mengambil semua properti dan semua metode dari kelas lain.

class LoginWindow(Screen):  
 email = ObjectProperty(None)  
 password = ObjectProperty(None)

class WindowManager(ScreenManager):  
 pass

Import file atau mengambil suatu data file .py yg berisi class dan object , sistem dan program yang di taruh ke suatu file lain

from kivy.app import App  
from kivy.lang import Builder  
from kivy.uix.screenmanager import ScreenManager, Screen  
from kivy.properties import ObjectProperty  
from kivy.uix.popup import Popup  
from kivy.uix.label import Label  
from database import DataBase