LAPORAN PROJECT PBO

(PYGAME & KIVY)

“Membuat Game Space Shuter dan Space Ballistic”



Oleh :

KHOIRUR ROSYIDI (160411100144)

ANDIKA ADITIA HARUN (160411100168)

**PRODI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**2018**

**KATA PENGANTAR**

Puji syukur atas kehadiran Tuhan Yang Maha Esa atas rahmat dan,petunjuk-Nya sehingga kami dapat menyelesaikan tugas Laporan Project Game kami dengan judul,“Game Sederhana dengan Pygame dan Kivy”, yang mana laporan ini disususn bertujuan untuk,memenuhi tugas akhir Mata Kuliah Pemrograman Bebasis Object (PBO) . Penulis menyadari bahwa masih banyak kekurangan dan keterbtasan,dalam penyajian data dalam Laporan ini. Oleh karena itu, kami mengharapkan, kritik dan saran yang membangun dari semua demi kesempurnaan Laporan ini. Semoga Laporan ini berguna dan dapat menambah pengetahuan kita semua.

Demikian laporan ini kami susun, apabila ada kata- kata yang kurang berkenan dan banyak terdapat kekurangan, kami mohon maaf yang sebesar-besarnya.

**Penulis**

**( Khoirur Rosyidi,**

**Andika Aditia H )**

**PENDAHULUAN**

**Latar Belakang**

Pygame adalah modul cross-platform dari Python dirancang untuk membuat game. Modulnya dirancang untuk menjadi sederhana, mudah digunakan, dan menyenangkan, kunci dari idologi Pygame. Pada post ini, saya akan tunjukkan bagaimana cara menggunakan Pygame dan berbagi tips dan bahan untuk mempelajarinya.

Python biasa disebut sebagai bahasa pemrograman terbaik untuk pertama kali belajar, dan banyak dipuji karena sintaks yang mudah dipelajari dan kurva belajar yang gradual. Untuk alasan ini, banyak programmer baru yang mulai belajar dengan Python.

Pygame mengembangkan Python, mengadopsi filosofi Python, dan ditargetkan agar mudah digunakan. Ditambah lagi, game developer baru dengan minim pengalaman bisa langsung menggunakan Pygame, secepat mereka mempelajari Python. Sedangkan Kivy adalah :

Kivy adalah GUI/NUI framework yang menggunakan bahasa pemrograman Python, dapat dijalankan pada Windows, Linux, OS X, Android dan iOS dengan menggunakan code yang sama dan menggunakan MIT license sehingga 100% free to use dan boleh dikomersialisasikan. Kivy didesain untuk membantu pembuatan dan pengembangan aplikasi cross-platform secara cepat.

Menurut saya, Kivy cocok untuk project kecil-kecilan dan pribadi, namun bisa dikembangkan ke arah komersil. Di blog ini saya akan share pengalaman belajar saya menggunakan Kivy dan mudah-mudahan kita bisa saling bertukar ilmu. Di tutorial selanjutnya saya akan menjelaskan meng-configure environment untuk men-develop Kivy beserta GUI editornya.

**Maksud Dan Tujuan**

1. a) Sebagai syarat Ujian Akhir Semester (UAS).
2. b) Meningkatkan pengetahuan dan keterampilan mahasiswa untuk menjadi tenaga kerja yang profesional.
3. c) Sebagai latihan bagi mahasiswa untuk terjun ke dunia nyata.

**DOKUMENTASI INSTALASI**

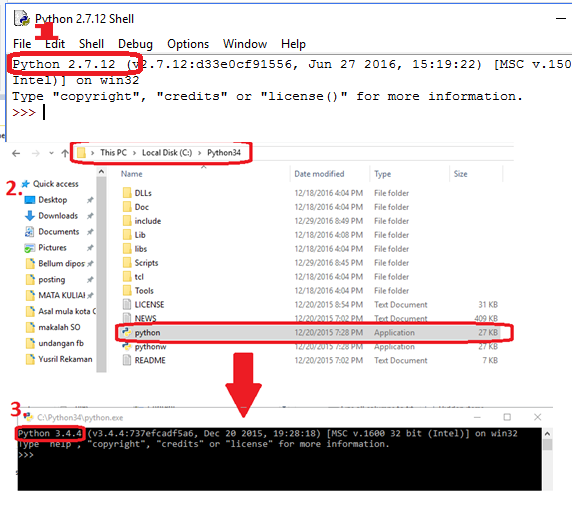
**PEMROGRAMAN BERORIENTASI OBJECT**

PyGame adalah program python yang berisi fungsi dan class yang dibutuhkan untuk membuat game.

Langkah 1 :  cek versi python.

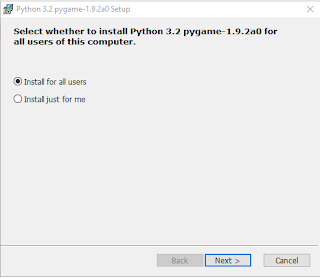
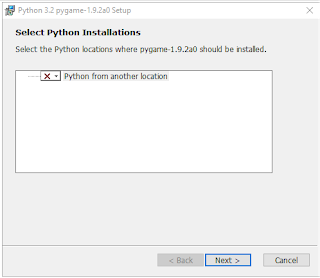
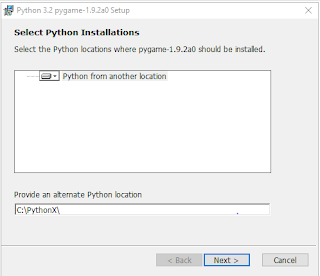
Saya anggap kalian sudah install python nya ya.. jika kalian belum install python nya ya… kalian bisa baca tutorial instalasi python nya DISINI YA temen - temen…. Oke kita balik lagi ke pembahasan utama kita di postingan kali ini ya… Nah langkah yang pertama ini, kalian harus mengetahui terlebih dahulu versi python yang terinstall di PC kalian dulu ya…. Bisa dengan cara membuka IDLE python nya seperti pada gambar no 1 di bawah ini. Atau bisa juga di C:/pythonxx/python.exe seperti pada gambar no 2 di bawah ini Maka akan muncul seperti pada gambar no 3.

Langkah 2 : Download file instalasi pygame nya.

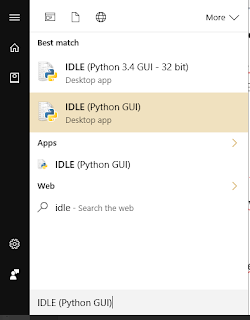
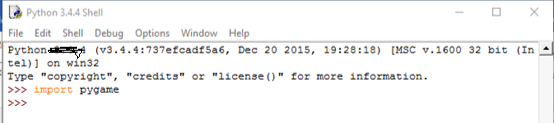
Setelah mengecek versi python nya, kalian harus pergi ke situs resminya dulu ya temen - temen.. Yang tentu saja untuk mendownload file instalasi pygame nya.. Situs resminya adalah --> <https://pygame.org/downloads>. Silahkan download sesuai versi python kalian ya.. Tentu saja pilih file instalasi dengan format “msi” agar mudah untuk di install nantinya...  silahkan pilih yang sesuai dengan versi python kalian sendiri ya temen – temen…  
Namun sayang sekali ya.. Pygame versi tertinggi yang tersedia dengan format “msi” hanya pada python 3.2 saja jika kalian memiliki versi yang lebih tinggi, kalian harus downgrade terlebih dahulu yaa...  namun jika kalian tidak mau mengganti versi python kalian, kalian bisa juga menginstall nya kok.. DAN TUTORIALNYA ADA DISINI… silahkan di baca ya…  
  
[[](http://4.bp.blogspot.com/-Q5BqrJF4rMo/WNFiw96dWiI/AAAAAAAAARg/GRSFr8XelFkcUClSZQrBMXENlFcvwhpIACK4B/s1600/1.png)](http://4.bp.blogspot.com/-Q5BqrJF4rMo/WNFiw96dWiI/AAAAAAAAARg/GRSFr8XelFkcUClSZQrBMXENlFcvwhpIACK4B/s1600/1.png)

**Langkah 3 : proses instalasi.**

Setelah kalian mendapatkan file nya, sekarang kita install seperti pada umumnya yaa.. Yaitu dengan cara :

1. klik 2 kali pada file nya... atau bisa kita katakan menjalankan file nya...
2. Kemudian lanjutkan ya.. klik “Run anyway”. Maka akan tampil window instalasi pada umumnya.. Seperti di bawah ini...  
     
   [](http://4.bp.blogspot.com/-1bwLFShxqlM/WNFjKUtjVZI/AAAAAAAAARo/sJCp74yHMPImrF9VipAlv4EKeMiAnejrgCK4B/s1600/2.png)
3. Nah klik next saja ya sampai kalian menemukan tampilan seperti di bawah ini yaa..  
     
   [](http://1.bp.blogspot.com/-ffUDXFxdy28/WNFjhjn_NjI/AAAAAAAAARw/lq5KJrNChjooOz21Vlzn8WTkyHZts4IpwCK4B/s1600/3.png)
4. Nah silahkan ganti pilihan instalasinya menjadi “specified hard disk” seperti di bawah ini.. Lalu pada bagian folder yang ada di pojok kiri bawah, kalian isi dengan lokasi instalasi file python kalian seperti yang ada pada gambar di atas ya temen - temen...  
     
   [](http://2.bp.blogspot.com/-iHydQ0cpLHQ/WNFjod-RdPI/AAAAAAAAAR4/uphszdAjmm0pYwoGD_K0Ae88-HmQT8-SgCK4B/s1600/4.png)
5. Nah setelah itu klik next dan tunggu hingga proses instalasinya selesai yaa... Hingga pygame nya siap untuk kalian gunakan...

**Langkah 4 : proses finishing dan testing.**

Nah setelah selesai, silahkan kalian cek apakah instalasinya berhasil atau tidak... Cara cek nya kalian bisa buka IDLE pada umumnya ya... Kalian bisa mecarinya di pencarian file seperti yang ada di bawah ini....  
  
[](http://2.bp.blogspot.com/-Ev8YmrywxRg/WNFj_FHSV1I/AAAAAAAAASA/Euu0B4fLtGwYSolITXDfdWNdJLVyrMs7QCK4B/s1600/6.png)  
  
Atau juga bisa menggunakan shell python nya... Yang biasanya berada di C:/python32/python  
Silahkan buka ya.. Lalu coba ketikkan code untuk mengimport pygame nya seperti di bawah ini..  
Import pygame  
  
[](http://3.bp.blogspot.com/-dTcx5AmFP5o/WNFkNGPLdhI/AAAAAAAAASI/lc7uDqXvtNYtzX-tps7bDbOGxlYcxPCXQCK4B/s1600/7.png)  
  
Nah jika kalian tidak mendapatkan error, maka pygame sukses kalian install pada pc windows kalian

**KIVY**

Kivy adalah GUI framework yang menggunakan bahasa pemrograman Python, Kivy didesain untuk membantu pembuatan dan pengembangan aplikasi cross-platform secara cepat.

Berikut adalah beberapa kelebihan Kivy

1. Cross-platform, code once run anywhere.
2. Support berbagai ragam input device dan juga multitouch support.
3. Menggunakan OpenGL ES sebagai graphics engine-nya.
4. Menggunakan bahasa Python, cocok bagi pemula seperti saya.
5. 3rd party Python library

**Cara install kivy**

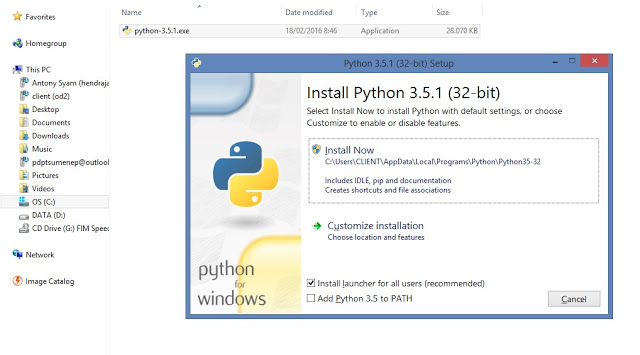
Langkah 1.

           mendownload aplikasi Python [di sini](https://www.python.org/downloads/), versi python ada 2 jenis yaitu versi python 3.5 dan versi python 2.7. Dalam proses instalasi ini saya menggunakan python 3.5.

[](https://2.bp.blogspot.com/-iHDC9nXsnG4/VsUhPJ5kWzI/AAAAAAAAAEg/i6fQx1rEWkE/s1600/2.jpg)

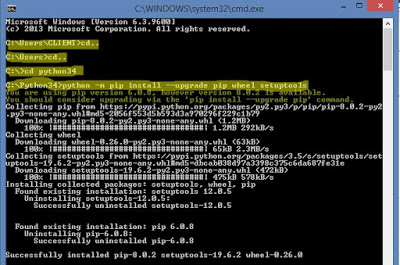
Langkah 2.

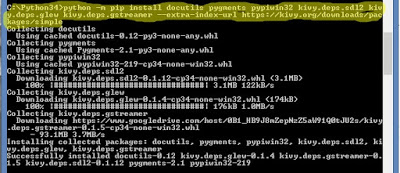
           setelah mendownload aplikasi python, langsung langkukan proses instalasi python dan tunggu proses instalasi python sampai selesai .

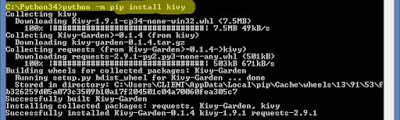
[[](https://2.bp.blogspot.com/-y2-8vP6_CAU/VsUjCprDefI/AAAAAAAAAEs/rJ-N72WLwyg/s1600/3.jpg)](https://2.bp.blogspot.com/-y2-8vP6_CAU/VsUjCprDefI/AAAAAAAAAEs/rJ-N72WLwyg/s1600/3.jpg)

Langkah 3.

           masuk ke CMD (*Command Prompt*) dengan cara menekan tombol Windows lalu ketik CMD atau dengan cara klik start, assesoris,  pilih commad promt. Maka akan masuk ke dalam kotak dialog Commad prompt. setelah masuk isikan coding sesuaikan seperti gambar yang ada di bawah ini.

[](https://3.bp.blogspot.com/-mQSCKJK3h0g/VsUoGsKw1iI/AAAAAAAAAFM/J5Y_N0ux6og/s1600/cdm+1.jpg)

[](https://3.bp.blogspot.com/-FE7aFKOiqA8/VsUoGi70hCI/AAAAAAAAAFQ/KSaxzTWSLcE/s1600/cdm+2.jpg)

[](https://1.bp.blogspot.com/-FgPbmrRTPGM/VsUoGs9ihHI/AAAAAAAAAFU/Nwx56UXlFdw/s1600/cdm+3.jpg)

            setelah proses instalasi selesai semua, maka bisa langsung di pakai.

**DOKUMENTASI SUMBER DAN HASIL RUN KIVY DAN PYGAME**

**PEMROGRAMAN BERORIENTASI OBJECT**

1. **Sumber**

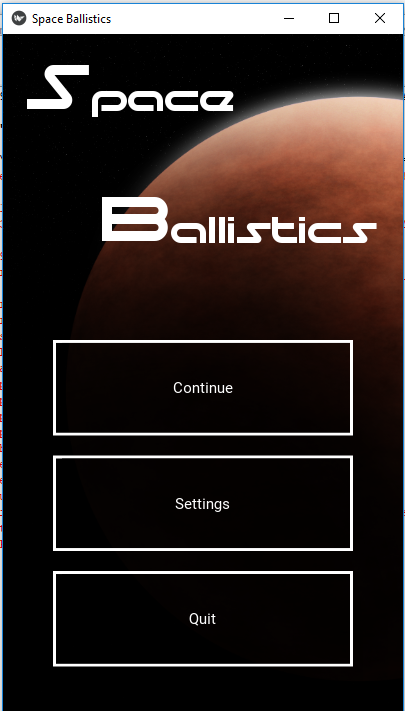
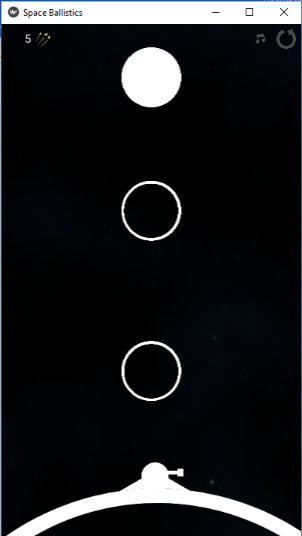
<https://github.com/Ross65536/kivy-ballistics>

<https://github.com/tasdikrahman/spaceShooter>

1. **Hasil Run**
2. **Pygame**



1. **Kiviy**



**DOKUMENTASI PYGAME**

**PEMROGRAMAN BERORIENTASI OBJECT**

**SPACESHOOTER**



Global Variable :

+SCREEN

Method and Class from pygame :

+ init()

+ sprite.Sprite

+ surface.Surface

+ sprite.Group

+ surface.Surface.fill

+ surface.Surface.rect

+ time.Clock

+ display.set\_mode

+ display.set\_caption

+ constants.\*

+ event.get

+ event.type vv

+ sprite.Group.draw

+ display.flip

+ time.Clock.tick

+ quit()

|  |
| --- |
| explotion |
| -center  -size |
| + update() |

Class dan Method for Snake Game

Class explotion:

Memiliki property :

* center letak koordinat image di posisi tengah
* size yaitu ukuran font dan gambar.

+ update = untuk mengupdate nilai waktu count down untuk gamenya bisa kita dapatkan dari fungsi

|  |
| --- |
| player |
| - |
| +update  +shoot  +powerup  +hide |

Class player

Method :

+ update =meng update power dan hidden jika power lebih besar dari 2 maka dikurangi 1.

+shoot adalah untuk menentukan pelura sesuia dengan power yang didapatkan

+powerup adalah menambahkan sub power

+hide adalah mengurangi nyawa ketika terkena batu

|  |
| --- |
| missile |
| -x  -y |
| +update |

Class missile

Memiliki property :

* x adalah posisi x rudal api
* y adalah posisi y rudal api

Method :

|  |
| --- |
| bullet |
| -x  -y |
| +update |

+ update untuk menggerkkan rudal api dan menghilangkan

Class bullet:

Memiliki property :

* x adalah posisi x awal peluru
* y adalah posisi y awal peluru

Method :

+update yaitu supaya peluluru muncul didepan pesawat.

|  |
| --- |
| mob |
| - |
| +rotate  +update |

Class mob:

Method :

+rotate untuk rotasi pergerakan musuh

+update untuk menentukan musuh muncul secara acak

|  |
| --- |
| pow |
| -center |
| +update |

Class pow:

Memiliki property :

* Center posisi turunnya shiled dan gun.

Method:

+ update menetukan kemunculan shile dan gun.

**DOKUMENTASI TUGAS KIVY**

**PEMROGRAMAN BERORIENTASI OBJECT**

**BALLISTIK**

Kivy.uix.widget.Widget

|  |
| --- |
| Planet |
| -\*\*kwargs |
|  |

|  |
| --- |
| Space\_Background |
| -main\_app  -background\_num  -randomf0to1 |
| +update\_pos  +rotate |

|  |
| --- |
| TurretGun |
|  |
| +create\_bullet\_count  +on\_touch\_down  +on\_touch\_up  +update |

|  |
| --- |
| BallBase |
| -root  -ball\_size  -ball\_position  -density  -\*\*kwargs |
| +update\_pos  +move  +colors\_and\_rotation |

|  |
| --- |
| BaseBullet |
| -pos  -speed  -\*\*kwargs |
| +update |

|  |
| --- |
| TurretBuilding |
|  |
|  |

|  |
| --- |
| TurretBarrel |
|  |
|  |

|  |
| --- |
| Planet\_Background |
| -main\_app  -background\_num  -randomf0to1 |
| +update\_pos |

|  |
| --- |
| GameWidget |
| -\*\*kwargs |
| +update\_balls\_pos  +update\_bullet\_graphic\_colors  +update  +update\_balls  +create\_ball  +schedule\_enemies |

kivy.app

.App

|  |
| --- |
| Gameapp |
|  |
| +build  +on\_pause  +update\_button\_colors  +destroy\_level  +create\_planet\_and\_turret  +create\_additional\_widgets\_on\_level\_creation  +create\_level  +hook\_keyboard  +replay\_button\_f  +music\_button\_f |

kivy.uix.togglebutton

ToggleButton

|  |
| --- |
| LevelSelectionButton |
| -\*\*kwargs |
| +on\_press\_f |

|  |
| --- |
| StartScreen |
|  |
|  |

|  |
| --- |
| LevelSelectionScreen |
|  |
|  |

kivy.uix.screenmanager

|  |
| --- |
| SettingsScreen |
|  |
|  |

|  |
| --- |
| LevelWonScreen |
|  |
|  |

|  |
| --- |
| LevelWonScreenEnd |
|  |
|  |

|  |
| --- |
| LevelLostScreen |
|  |
|  |

|  |
| --- |
| LevelLostScreen2 |
|  |
|  |

#CLASS AND METHOD DI ballistik

1. Class SpaceBaground

Class SpaceBaground class yang memproses semua yang mengenai Baground pada widget

Merupakan turunan dari class widget uix

Method:

+update\_pos update screen size

+rotate menentukan rotasi

1. PlanetBackground

Class PlanetBackground class yang memproses semua yang mengenai Baground planet pada widget

Merupakan turunan dari class widget uix

1. TurretBarrel warna dan radius barrel

Method:

+create\_bullet\_count menentukan perhitungan count radius

+on\_touch\_down gerak barrel

+on\_touch\_up gerak barrel

+update

1. Planet menentukan radius setiap planet pada widget
2. BallBase menentukan ukaran dan warna pada ball base
3. TurretGun menentukan posisi dan gerak pada gun
4. GameWidget memnentukan ukuran game, posisi game dan semua yang berhubungan dengan game

Method:

+schedule\_enemies

+update\_balls\_pos menentukan posisi ball

+update\_bullet\_graphic\_colors mengubah warna grafik pada game dan ball

+update

+update\_balls

+create\_ball

1. Gameapp

Adalah class yang berhubangan di setiap bagian game seperti level,nyawa,dll

Method:

+build

+on\_pause membuat pause saat game berjalan

+update\_button\_colors menentukan warna setiap level

+destroy\_level level berkulang saat gak mengenai bidikan

+create\_planet\_and\_turret membuat planet dan turret

+create\_additional\_widgets\_on\_level\_creation setting replay butto music button dll

+create\_level menentukan apa yang dilakukan saat berada di posisi level tertentu

Dengan Code sebagai berikut :

* **Pygame**

from \_\_future\_\_ import division

import pygame

import random

from os import path

## folder assets

img\_dir = path.join(path.dirname(\_\_file\_\_), 'assets')

sound\_folder = path.join(path.dirname(\_\_file\_\_), 'sounds')

WIDTH = 480

HEIGHT = 600

FPS = 60

POWERUP\_TIME = 5000

BAR\_LENGTH = 100

BAR\_HEIGHT = 10

# warna

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)

YELLOW = (255, 255, 0)

pygame.init()

pygame.mixer.init() ## suara

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Space Shooter")

clock = pygame.time.Clock()

font\_name = pygame.font.match\_font('arial')

def main\_menu():

global screen

menu\_song = pygame.mixer.music.load(path.join(sound\_folder, "menu.ogg"))

pygame.mixer.music.play(-1)

title = pygame.image.load(path.join(img\_dir, "main.png")).convert()

title = pygame.transform.scale(title, (WIDTH, HEIGHT), screen)

screen.blit(title, (0,0))

pygame.display.update()

while True:

ev = pygame.event.poll()

if ev.type == pygame.KEYDOWN:

if ev.key == pygame.K\_RETURN:

break

elif ev.key == pygame.K\_q:

pygame.quit()

quit()

elif ev.type == pygame.QUIT:

pygame.quit()

quit()

else:

draw\_text(screen, "Tekan [ENTER] untuk Mulai", 30, WIDTH/2, HEIGHT/2)

draw\_text(screen, "atau [Q] untuk Keluar", 30, WIDTH/2, (HEIGHT/2)+40)

pygame.display.update()

#pygame.mixer.music.stop()

ready = pygame.mixer.Sound(path.join(sound\_folder,'getready.ogg'))

ready.play()

screen.fill(BLACK)

draw\_text(screen, "SUDAH SIAP!", 40, WIDTH/2, HEIGHT/2)

pygame.display.update()

def draw\_text(surf, text, size, x, y):

## memilih font untuk menampilkan score

font = pygame.font.Font(font\_name, size)

text\_surface = font.render(text, True, WHITE)

text\_rect = text\_surface.get\_rect()

text\_rect.midtop = (x, y)

surf.blit(text\_surface, text\_rect)

def draw\_shield\_bar(surf, x, y, pct):

# if pct < 0:

# pct = 0

pct = max(pct, 0)

fill = (pct / 100) \* BAR\_LENGTH

outline\_rect = pygame.Rect(x, y, BAR\_LENGTH, BAR\_HEIGHT)

fill\_rect = pygame.Rect(x, y, fill, BAR\_HEIGHT)

pygame.draw.rect(surf, GREEN, fill\_rect)

pygame.draw.rect(surf, WHITE, outline\_rect, 2)

def draw\_lives(surf, x, y, lives, img):

for i in range(lives):

img\_rect= img.get\_rect()

img\_rect.x = x + 30 \* i

img\_rect.y = y

surf.blit(img, img\_rect)

def newmob():

mob\_element = Mob()

all\_sprites.add(mob\_element)

mobs.add(mob\_element)

class Explosion(pygame.sprite.Sprite):

def \_\_init\_\_(self, center, size):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.size = size

self.image = explosion\_anim[self.size][0]

self.rect = self.image.get\_rect()

self.rect.center = center

self.frame = 0

self.last\_update = pygame.time.get\_ticks()

self.frame\_rate = 75

def update(self):

now = pygame.time.get\_ticks()

if now - self.last\_update > self.frame\_rate:

self.last\_update = now

self.frame += 1

if self.frame == len(explosion\_anim[self.size]):

self.kill()

else:

center = self.rect.center

self.image = explosion\_anim[self.size][self.frame]

self.rect = self.image.get\_rect()

self.rect.center = center

class Player(pygame.sprite.Sprite):

def \_\_init\_\_(self):

pygame.sprite.Sprite.\_\_init\_\_(self)

## skala gambar pemain

self.image = pygame.transform.scale(player\_img, (50, 38))

self.image.set\_colorkey(BLACK)

self.rect = self.image.get\_rect()

self.radius = 20

self.rect.centerx = WIDTH / 2

self.rect.bottom = HEIGHT - 10

self.speedx = 0

self.speedy = 0

self.shield = 100

self.shoot\_delay = 250

self.last\_shot = pygame.time.get\_ticks()

self.lives = 4

self.hidden = False

self.hide\_timer = pygame.time.get\_ticks()

self.power = 1

self.power\_timer = pygame.time.get\_ticks()

def update(self):

if self.power >=2 and pygame.time.get\_ticks() - self.power\_time > POWERUP\_TIME:

self.power -= 1

self.power\_time = pygame.time.get\_ticks()

if self.hidden and pygame.time.get\_ticks() - self.hide\_timer > 1000:

self.hidden = False

self.rect.centerx = WIDTH / 2

self.rect.bottom = HEIGHT - 30

self.speedx = 0

keystate = pygame.key.get\_pressed()

if keystate[pygame.K\_LEFT]:

self.speedx = -5

elif keystate[pygame.K\_RIGHT]:

self.speedx = 5

if keystate[pygame.K\_SPACE]:

self.shoot()

if self.rect.right > WIDTH:

self.rect.right = WIDTH

if self.rect.left < 0:

self.rect.left = 0

self.rect.x += self.speedx

def shoot(self):

## memberi tahu peluru

now = pygame.time.get\_ticks()

if now - self.last\_shot > self.shoot\_delay:

self.last\_shot = now

if self.power == 1:

bullet = Bullet(self.rect.centerx, self.rect.top)

all\_sprites.add(bullet)

bullets.add(bullet)

shooting\_sound.play()

if self.power == 2:

bullet1 = Bullet(self.rect.left, self.rect.centery)

bullet2 = Bullet(self.rect.right, self.rect.centery)

all\_sprites.add(bullet1)

all\_sprites.add(bullet2)

bullets.add(bullet1)

bullets.add(bullet2)

shooting\_sound.play()

""" MOAR POWAH """

if self.power == 3:

bullet1 = Bullet(self.rect.left, self.rect.centery)

bullet2 = Bullet(self.rect.right, self.rect.centery)

missile1 = Missile(self.rect.centerx, self.rect.top)

all\_sprites.add(bullet1)

all\_sprites.add(bullet2)

all\_sprites.add(missile1)

bullets.add(bullet1)

bullets.add(bullet2)

bullets.add(missile1)

shooting\_sound.play()

missile\_sound.play()

def powerup(self):

self.power += 1

self.power\_time = pygame.time.get\_ticks()

def hide(self):

self.hidden = True

self.hide\_timer = pygame.time.get\_ticks()

self.rect.center = (WIDTH / 2, HEIGHT + 200)

# dfinisikan musuh

class Mob(pygame.sprite.Sprite):

def \_\_init\_\_(self):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image\_orig = random.choice(meteor\_images)

self.image\_orig.set\_colorkey(BLACK)

self.image = self.image\_orig.copy()

self.rect = self.image.get\_rect()

self.radius = int(self.rect.width \*.90 / 2)

self.rect.x = random.randrange(0, WIDTH - self.rect.width)

self.rect.y = random.randrange(-150, -100)

self.speedy = random.randrange(5, 20)

self.speedx = random.randrange(-3, 3)

self.rotation = 0

self.rotation\_speed = random.randrange(-8, 8)

self.last\_update = pygame.time.get\_ticks()

def rotate(self):

time\_now = pygame.time.get\_ticks()

if time\_now - self.last\_update > 50:

self.last\_update = time\_now

self.rotation = (self.rotation + self.rotation\_speed) % 360

new\_image = pygame.transform.rotate(self.image\_orig, self.rotation)

old\_center = self.rect.center

self.image = new\_image

self.rect = self.image.get\_rect()

self.rect.center = old\_center

def update(self):

self.rotate()

self.rect.x += self.speedx

self.rect.y += self.speedy

if (self.rect.top > HEIGHT + 10) or (self.rect.left < -25) or (self.rect.right > WIDTH + 20):

self.rect.x = random.randrange(0, WIDTH - self.rect.width)

self.rect.y = random.randrange(-100, -40)

self.speedy = random.randrange(1, 8)

class Pow(pygame.sprite.Sprite):

def \_\_init\_\_(self, center):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.type = random.choice(['shield', 'gun'])

self.image = powerup\_images[self.type]

self.image.set\_colorkey(BLACK)

self.rect = self.image.get\_rect()

self.rect.center = center

self.speedy = 2

def update(self):

self.rect.y += self.speedy

if self.rect.top > HEIGHT:

self.kill()

class Bullet(pygame.sprite.Sprite):

def \_\_init\_\_(self, x, y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image = bullet\_img

self.image.set\_colorkey(BLACK)

self.rect = self.image.get\_rect()

self.rect.bottom = y

self.rect.centerx = x

self.speedy = -10

def update(self):

"""should spawn right in front of the player"""

self.rect.y += self.speedy

if self.rect.bottom < 0:

self.kill()

## Rudal Api

class Missile(pygame.sprite.Sprite):

def \_\_init\_\_(self, x, y):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image = missile\_img

self.image.set\_colorkey(BLACK)

self.rect = self.image.get\_rect()

self.rect.bottom = y

self.rect.centerx = x

self.speedy = -10

def update(self):

self.rect.y += self.speedy

if self.rect.bottom < 0:

self.kill()

## muat semua gambar game

background = pygame.image.load(path.join(img\_dir, 'starfield.png')).convert()

background\_rect = background.get\_rect()

player\_img = pygame.image.load(path.join(img\_dir, 'playerShip1\_orange.png')).convert()

player\_mini\_img = pygame.transform.scale(player\_img, (25, 19))

player\_mini\_img.set\_colorkey(BLACK)

bullet\_img = pygame.image.load(path.join(img\_dir, 'laserRed16.png')).convert()

missile\_img= pygame.image.load(path.join(img\_dir, 'missile.png')).convert\_alpha()

# meteor\_img = pygame.image.load(path.join(img\_dir, 'meteorBrown\_med1.png')).convert()

meteor\_images = []

meteor\_list = [

'meteorBrown\_big1.png',

'meteorBrown\_big2.png',

'meteorBrown\_med1.png',

'meteorBrown\_med3.png',

'meteorBrown\_small1.png',

'meteorBrown\_small2.png',

'meteorBrown\_tiny1.png'

]

for image in meteor\_list:

meteor\_images.append(pygame.image.load(path.join(img\_dir, image)).convert())

## ledakan meteor

explosion\_anim = {}

explosion\_anim['lg'] = []

explosion\_anim['sm'] = []

explosion\_anim['player'] = []

for i in range(9):

filename = 'regularExplosion0{}.png'.format(i)

img = pygame.image.load(path.join(img\_dir, filename)).convert()

img.set\_colorkey(BLACK)

## bangkitkan ledakan

img\_lg = pygame.transform.scale(img, (75, 75))

explosion\_anim['lg'].append(img\_lg)

img\_sm = pygame.transform.scale(img, (32, 32))

explosion\_anim['sm'].append(img\_sm)

## ledakan pemain

filename = 'sonicExplosion0{}.png'.format(i)

img = pygame.image.load(path.join(img\_dir, filename)).convert()

img.set\_colorkey(BLACK)

explosion\_anim['player'].append(img)

powerup\_images = {}

powerup\_images['shield'] = pygame.image.load(path.join(img\_dir, 'shield\_gold.png')).convert()

powerup\_images['gun'] = pygame.image.load(path.join(img\_dir, 'bolt\_gold.png')).convert()

# Load all game sounds

shooting\_sound = pygame.mixer.Sound(path.join(sound\_folder, 'pew.wav'))

missile\_sound = pygame.mixer.Sound(path.join(sound\_folder, 'rocket.ogg'))

expl\_sounds = []

for sound in ['expl3.wav', 'expl6.wav']:

expl\_sounds.append(pygame.mixer.Sound(path.join(sound\_folder, sound)))

## main background music

#pygame.mixer.music.load(path.join(sound\_folder, 'tgfcoder-FrozenJam-SeamlessLoop.ogg'))

pygame.mixer.music.set\_volume(0.2)

player\_die\_sound = pygame.mixer.Sound(path.join(sound\_folder, 'rumble1.ogg'))

###################################################

# Error :

# TypeError: play() takes no keyword arguments

#pygame.mixer.music.play()

running = True

menu\_display = True

while running:

if menu\_display:

main\_menu()

pygame.time.wait(3000)

pygame.mixer.music.stop()

pygame.mixer.music.load(path.join(sound\_folder, 'tgfcoder-FrozenJam-SeamlessLoop.ogg'))

pygame.mixer.music.play(-1)

menu\_display = False

all\_sprites = pygame.sprite.Group()

player = Player()

all\_sprites.add(player)

mobs = pygame.sprite.Group()

for i in range(8):

# mob\_element = Mob()

# all\_sprites.add(mob\_element)

# mobs.add(mob\_element)

newmob()

bullets = pygame.sprite.Group()

powerups = pygame.sprite.Group()

score = 0

clock.tick(FPS)

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_ESCAPE:

running = False

all\_sprites.update()

hits = pygame.sprite.groupcollide(mobs, bullets, True, True)

for hit in hits:

score += 50 - hit.radius

random.choice(expl\_sounds).play()

# m = Mob()

# all\_sprites.add(m)

# mobs.add(m)

expl = Explosion(hit.rect.center, 'lg')

all\_sprites.add(expl)

if random.random() > 0.9:

pow = Pow(hit.rect.center)

all\_sprites.add(pow)

powerups.add(pow)

newmob()

hits = pygame.sprite.spritecollide(player, mobs, True, pygame.sprite.collide\_circle)

for hit in hits:

player.shield -= hit.radius \* 2

expl = Explosion(hit.rect.center, 'sm')

all\_sprites.add(expl)

newmob()

if player.shield <= 0:

player\_die\_sound.play()

death\_explosion = Explosion(player.rect.center, 'player')

all\_sprites.add(death\_explosion)

# running = False ## GAME OVER 3:D

player.hide()

player.lives -= 1

player.shield = 100

hits = pygame.sprite.spritecollide(player, powerups, True)

for hit in hits:

if hit.type == 'shield':

player.shield += random.randrange(10, 30)

if player.shield >= 100:

player.shield = 100

if hit.type == 'gun':

player.powerup()

## jika pemain mati maka perminan selesai

if player.lives == 0 and not death\_explosion.alive():

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

menu\_song = pygame.mixer.music.load(path.join(sound\_folder, "menu.ogg"))

pygame.mixer.music.play(-1)

title = pygame.image.load(path.join(img\_dir, "main.png")).convert()

title = pygame.transform.scale(title, (WIDTH, HEIGHT), screen)

screen.blit(title, (0,0))

pygame.display.update()

while True:

ev = pygame.event.poll()

if ev.type == pygame.KEYDOWN:

if ev.key == pygame.K\_q:

pygame.quit()

quit()

elif ev.type == pygame.QUIT:

pygame.quit()

quit()

else:

draw\_text(screen, "GAME OVER", 30, WIDTH/2, HEIGHT/2)

draw\_text(screen, "atau [Q] untuk Keluar", 30, WIDTH/2, (HEIGHT/2)+40)

pygame.display.update()

# menu\_display = True

# pygame.display.update()

screen.fill(BLACK)

screen.blit(background, background\_rect)

all\_sprites.draw(screen)

draw\_text(screen, str(score), 18, WIDTH / 2, 10)

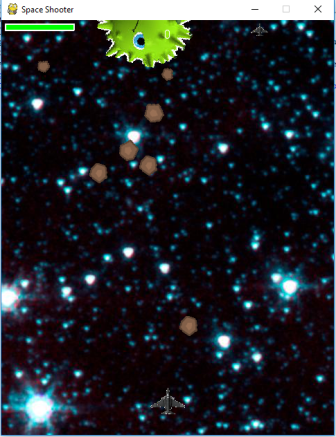
draw\_shield\_bar(screen, 5, 5, player.shield)

draw\_lives(screen, WIDTH - 120, 4, player.lives, player\_mini\_img)

pygame.display.flip()

pygame.quit()

* **Dokumentasi**



* **KIVY :**

import kivy

import os

import random

kivy.require('1.9.0')

from kivy.app import App

from kivy.uix.button import Button

from kivy.uix.togglebutton import ToggleButton

from kivy.uix.widget import Widget

from kivy.properties import ReferenceListProperty

from kivy.properties import ObjectProperty

from kivy.graphics import Color

from kivy.graphics import Rectangle

from kivy.graphics import Ellipse

from kivy.graphics import PopMatrix

from kivy.graphics import PushMatrix

from kivy.graphics import Rotate

from kivy.clock import Clock

from kivy.config import Config

from kivy.vector import Vector

from kivy.uix.screenmanager import ScreenManager, Screen

from kivy.base import EventLoop

from functools import partial

from random import random, randint

import itertools

import math

from kivy.uix.screenmanager import FadeTransition, FallOutTransition

#from kivy.core.audio import SoundLoader

#from kivy.core.audio.audio\_sdl2 import SoundSDL2

from kivy.uix.label import Label

from kivy.uix.popup import Popup

from kivy.core.text import LabelBase

Config.set('graphics', 'width', '400')

Config.set('graphics', 'height', '711')

# settings, level selection : Rabbid-Highway-Sign

KIVY\_FONTS = [

{

"name": "moonhouse",

"fn\_regular": "Resources/fonts/moonhouse.ttf"

}

]

for font in KIVY\_FONTS:

LabelBase.register(\*\*font)

class Space\_Background (Widget):

rect=None

main\_app= None

background\_num= None

def \_\_init\_\_ (self, main\_app, background\_num= None, randomf0to1=None):

super(Space\_Background, self).\_\_init\_\_()

self.main\_app=main\_app

if background\_num==None:

background\_num= randint(1,main\_app.number\_of\_backgrounds)

if randomf0to1==None:

randomf0to1=random()

self.background\_num=background\_num

with self.canvas:

self.rect=Rectangle(source= 'Resources/backgrounds/' + str(background\_num) + '.png' , pos= (-0.25 \* main\_app.screen\_manager.height \* randomf0to1 ,0) , size= (main\_app.screen\_manager.height , main\_app.screen\_manager.height))

def update\_pos (self, a2, screen\_size):

self.rect.size= (screen\_size[1] , screen\_size[1])

def rotate (self):

rand=self.background\_num

while rand== self.background\_num:

rand=randint(1,self.main\_app.number\_of\_backgrounds)

randomf0to1=random()

self.background\_num=rand

self.rect.source= 'Resources/backgrounds/' + str(rand) + '.png'

self.rect.pos= (-0.25 \* self.main\_app.screen\_manager.height \* randomf0to1 , 0)

class Planet\_Background (Widget):

rect=None

main\_app= None

def \_\_init\_\_ (self, main\_app, background\_num= None, randomf0to1=0):

super(Planet\_Background, self).\_\_init\_\_()

self.main\_app=main\_app

with self.canvas:

self.rect=Rectangle(source= 'Resources/backgrounds/planet\_background.png' , pos= (0,0) , size= (main\_app.screen\_manager.height , main\_app.screen\_manager.height))

def update\_pos (self, a2, screen\_size):

self.rect.size= (screen\_size[1] , screen\_size[1])

class StartScreen(Screen):

game\_button= ObjectProperty(None)

pass

class LevelSelectionScreen(Screen):

gridlayout=ObjectProperty(None)

pass

class SettingsScreen(Screen):

checkbox\_SFX=ObjectProperty(None)

checkbox\_Music=ObjectProperty(None)

pass

class GameScreen(Screen):

pass

class LevelWonScreen(Screen):

pass

class LevelWonScreenEnd(Screen):

pass

class LevelLostScreen(Screen):

pass

class LevelLostScreen2(LevelLostScreen):

lost\_label= None

pass

class LevelSelectionButton(Button):

main\_app= ObjectProperty(None)

level= ObjectProperty(None)

text= ObjectProperty(None)

def \_\_init\_\_ (self, \*\*kwargs):

super(LevelSelectionButton, self).\_\_init\_\_(\*\*kwargs)

self.bind(on\_press=self.on\_press\_f)

def on\_press\_f(self, instance):

if self.main\_app.beat\_to\_the\_level >= self.level:

self.main\_app.game\_widget.current\_level=self.level

self.main\_app.create\_level()

self.main\_app.screen\_game.space\_background.rotate()

self.main\_app.screen\_manager.current = 'game\_screen'

class Widget(Widget):

bbase\_enemy\_ball=False

bbase\_objective\_ball=False

class BaseBullet(Widget):

speed=[0,0]

ellip=None

def \_\_init\_\_(self,pos,speed, \*\*kwargs):

super(BaseBullet, self).\_\_init\_\_(\*\*kwargs)

self.pos=pos

self.speed=speed

def update(self, root, dt):

self.pos[0]=self.pos[0]+dt\*self.speed[0]

self.pos[1]=self.pos[1]+dt\*self.speed[1]

#self.draw\_instruction.pos=self.pos

ar=[0, 0]

for ball in root.balls\_list:

dy=ball[0][1]-self.pos[1]

dx=ball[0][0]-self.pos[0]

r2=dx\*\*2+dy\*\*2

r= math.sqrt(r2)

if (r<=ball[1]):

return 1

ang=math.atan2(dy,dx)

a = ( root.gravity\_constant \* root.bullet\_gravity\_constant \* ball[2] / float(r2) )

ar[0] = ar[0] + a \* math.cos(ang)

ar[1] = ar[1] + a \* math.sin(ang)

for ball in root.balls\_objective\_list:

dy=ball[0][1]-self.pos[1]

dx=ball[0][0]-self.pos[0]

r2=dx\*\*2+dy\*\*2

r= math.sqrt(r2)

if (r<=ball[1]):

root.remove\_widget(ball[4])

root.balls\_objective\_list.remove(ball)

return 2

ang=math.atan2(dy,dx)

a = ( root.gravity\_constant \* root.bullet\_gravity\_constant \* ball[3] \* ball[2] / float(r2) )

ar[0] = ar[0] + a \* math.cos(ang)

ar[1] = ar[1] + a \* math.sin(ang)

# [ [ [center\_x, center\_y], radius, mass, density, widget\_ref ] , ...]

self.speed=[self.speed[0] + ar[0] \* dt , self.speed[1] + ar[1] \* dt ]

if (self.pos[1]>1.5 \* root.height) or (self.pos[1]< -0.5 \* root.height):

return 1

elif self.pos[0]<-0.5 \* root.width or self.pos[0]>root.width \* 1.5: # TODO: Physics!

return 1

return 0

class Bullet(BaseBullet):

pass

class HeavyBullet(BaseBullet):

def update(self, root, dt):

self.pos[0]=self.pos[0]+dt\*self.speed[0]

self.pos[1]=self.pos[1]+dt\*self.speed[1]

#self.draw\_instruction.pos=self.pos

ar=[0, 0]

for ball in root.balls\_list:

dy=-1\*ball[0][1]+self.pos[1]

dx=-1\*ball[0][0]+self.pos[0]

r2=dx\*\*2+dy\*\*2

r= math.sqrt(r2)

ang=math.atan2(dy,dx)

if (r<=ball[1]):

ball[4].bmove=True

ball[4].brotate=True

# TODO: rotation maths

ball[4].brotate=False

vang=math.atan2(self.speed[1], self.speed[0])

dang=ang-vang

cdang=math.cos(dang)

dvCM=math.sqrt((self.speed[1]\*\*2+self.speed[0]\*\*2) \* 1.0 \* root.heavy\_bullet\_constant \* (cdang \*\* 2) / ball[2])

ball[4].speed[0]=ball[4].speed[0] - dvCM \* math.cos(ang)

ball[4].speed[1]=ball[4].speed[1] - dvCM \* math.sin(ang)

return 1

a = ( root.gravity\_constant \* root.bullet\_gravity\_constant \* ball[3] \* ball[2] / float(r2) )

ar[0] = ar[0] - a \* math.cos(ang)

ar[1] = ar[1] - a \* math.sin(ang)

for ball in root.balls\_objective\_list:

dy=-1\*ball[0][1]+self.pos[1] # change to center\_x

dx=-1\*ball[0][0]+self.pos[0]

r2=dx\*\*2+dy\*\*2

ang=math.atan2(dy,dx)

r= math.sqrt(r2)

if (r<=ball[1]):

ball[4].bmove=True

ball[4].brotate=True

# TODO: rotation maths

ball[4].brotate=False

vang=math.atan2(self.speed[1], self.speed[0])

dang=ang-vang

cdang=math.cos(dang)

dvCM=math.sqrt((self.speed[1]\*\*2+self.speed[0]\*\*2) \* 1.0 \* root.heavy\_bullet\_constant \* (cdang \*\* 2) / ball[2])

ball[4].speed[0]=ball[4].speed[0] - dvCM \* math.cos(ang)

ball[4].speed[1]=ball[4].speed[1] - dvCM \* math.sin(ang)

return 1

a = ( root.gravity\_constant \* root.bullet\_gravity\_constant \* ball[2] / float(r2) )

ar[0] = ar[0] - a \* math.cos(ang)

ar[1] = ar[1] - a \* math.sin(ang)

# [ [ [center\_x, center\_y], radius, mass, density, widget\_ref ] , ...]

self.speed=[self.speed[0] + ar[0] \* dt , self.speed[1] + ar[1] \* dt ]

if (self.pos[1]>1.2 \* root.height) or (self.pos[1]< -0.2 \* root.height):

return 1

elif self.pos[0]<-0.5 \* root.width or self.pos[0]>root.width \* 1.5: # TODO: Physics!

return 1

return 0

class StrangeMatterBullet(BaseBullet):

def update(self, root, dt):

self.pos[0]=self.pos[0]+dt\*self.speed[0]

self.pos[1]=self.pos[1]+dt\*self.speed[1]

#self.draw\_instruction.pos=self.pos

ar=[0, 0]

for ball in root.balls\_list:

dy=-1\*ball[0][1]+self.pos[1]

dx=-1\*ball[0][0]+self.pos[0]

r2=dx\*\*2+dy\*\*2

r= math.sqrt(r2)

if (r<=ball[1]):

root.remove\_widget(ball[4])

root.balls\_list.remove(ball)

return 1

for ball in root.balls\_objective\_list:

dy=-1\*ball[0][1]+self.pos[1] # change to center\_x

dx=-1\*ball[0][0]+self.pos[0]

r2=dx\*\*2+dy\*\*2

r= math.sqrt(r2)

if (r<=ball[1]):

root.remove\_widget(ball[4])

root.balls\_objective\_list.remove(ball)

return 2

# [ [ [center\_x, center\_y], radius, mass, density, widget\_ref ] , ...]

if (self.pos[1]>1.2 \* root.height) or (self.pos[1]< -0.2 \* root.height):

return 1

elif self.pos[0]<-0.5 \* root.width or self.pos[0]>root.width \* 1.5: # TODO: Physics!

return 1

return 0

class Planet(Widget):

benemy=False

bupdate= False

radius= ObjectProperty(None)

mass= ObjectProperty(None)

'''

def \_\_init\_\_ (self, \*\*kwargs):

super(Planet, self).\_\_init\_\_(\*\*kwargs)

self.mass=self.radius\*self.radius\*math.pi

'''

class TurretBarrel(Widget):

pass

class TurretBuilding(Widget):

benemy=False

bupdate=False

pass

class TurretGun(Widget):

benemy=False

bupdate=True

bullet\_speed\_factor=1.2

bullet\_damage\_factor=2.0

num\_bullets\_be\_destroy=3

num\_heavy\_bullets\_to\_destroy=0

num\_bulletsy\_to\_fire=3

num\_heavy\_bullets\_to\_fire=0

num\_strange\_bullets\_to\_fire=0

num\_strange\_bullets\_to\_destroy=0

current\_ammo\_type=0 # 0 - normall || 1 - heavy

num\_ammo\_types=3

bshoot=False

root\_widget=ObjectProperty(None)

rotate\_instruction= ObjectProperty(None)

def create\_bullet\_count(self, num\_bullets\_for\_level=3, num\_heavy\_bullets=0, num\_strange\_bullets=0):

self.num\_bulletsy\_to\_fire=num\_bullets\_for\_level

self.num\_bullets\_be\_destroy=num\_bullets\_for\_level

self.num\_heavy\_bullets\_to\_fire=num\_heavy\_bullets

self.num\_heavy\_bullets\_to\_destroy=num\_heavy\_bullets

self.num\_strange\_bullets\_to\_fire=num\_strange\_bullets

self.num\_strange\_bullets\_to\_destroy=num\_strange\_bullets

def on\_touch\_down(self, touch):

if touch.y < self.y - self.barrel\_length/2.0:

if touch.is\_double\_tap:

# TODO:

if not (self.num\_bulletsy\_to\_fire==0 and self.num\_heavy\_bullets\_to\_fire==0 and self.num\_strange\_bullets\_to\_fire==0):

self.current\_ammo\_type = (1 + self.current\_ammo\_type) % self.num\_ammo\_types

brotate=True

while brotate:

if self.current\_ammo\_type==0:

if self.num\_bulletsy\_to\_fire==0:

self.current\_ammo\_type = (1 + self.current\_ammo\_type) % self.num\_ammo\_types

else:

brotate=False

elif self.current\_ammo\_type==1:

if self.num\_heavy\_bullets\_to\_fire==0:

self.current\_ammo\_type = (1 + self.current\_ammo\_type) % self.num\_ammo\_types

else:

brotate=False

elif self.current\_ammo\_type==2:

if self.num\_strange\_bullets\_to\_fire==0:

self.current\_ammo\_type = (1 + self.current\_ammo\_type) % self.num\_ammo\_types

else:

brotate=False

self.root\_widget.update\_bullet\_graphic\_colors()

#print "ammo type "+ str(self.current\_ammo\_type)

elif touch.y >= self.y - self.barrel\_length/2.0:

self.touch=touch

self.bshoot=True

if self.touch.x-self.x==0:

self.ang= math.pi/2

else:

frac=0.0

self.ang=math.atan2(self.touch.y\*1.0-self.y , self.touch.x\*1.0-self.x)

if self.ang < math.pi\*frac and self.ang > -1\*math.pi/2:

self.ang=math.pi\*frac

elif self.ang > math.pi\*(1-frac) or self.ang < -1\*math.pi/2:

self.ang=math.pi\*(1-frac)

# ------------

self.rotate\_instruction.angle= self.ang \* 180 / math.pi

print ("{} - {}".format( touch.x\*1.0/self.root\_widget.width , touch.y\*1.0/self.root\_widget.height))

return True

def on\_touch\_up(self, touch):

if self.bshoot:

self.bshoot=False

if (self.current\_ammo\_type==0 and self.num\_bulletsy\_to\_fire>0) or (self.current\_ammo\_type==1 and self.num\_heavy\_bullets\_to\_fire>0) or (self.current\_ammo\_type==2 and self.num\_strange\_bullets\_to\_fire>0):

#!!

factor=1

if self.current\_ammo\_type==2:

factor=1.2

speed=self.bullet\_speed\_factor\*self.root\_widget.width \* factor

s=math.sin(self.ang)

c=math.cos(self.ang)

root=self.root\_widget

if self.current\_ammo\_type==0:

bull=Bullet([self.x+c\*self.barrel\_length,self.y+s\*self.barrel\_length], [speed\*c,speed\*s])

self.num\_bulletsy\_to\_fire-=1

self.root\_widget.normal\_bullets\_label.text=str(self.num\_bulletsy\_to\_fire)

'''if root.main\_app.bMute\_SFX==False:

root.SFX\_small\_Bullets.play()

else:

if root.SFX\_small\_Bullets.state=='play':

root.SFX\_small\_Bullets.stop()'''

elif self.current\_ammo\_type==1:

bull=HeavyBullet([self.x+c\*self.barrel\_length,self.y+s\*self.barrel\_length], [speed\*c,speed\*s])

self.num\_heavy\_bullets\_to\_fire-=1

self.root\_widget.heavy\_bullets\_label.text=str(self.num\_heavy\_bullets\_to\_fire)

'''if root.main\_app.bMute\_SFX==False:

root.SFX\_Heavy\_Bullets.play()

else:

if root.SFX\_Heavy\_Bullets.state=='play':

root.SFX\_Heavy\_Bullets.stop()'''

elif self.current\_ammo\_type==2:

bull=StrangeMatterBullet([self.x+c\*self.barrel\_length,self.y+s\*self.barrel\_length], [speed\*c,speed\*s])

self.num\_strange\_bullets\_to\_fire-=1

self.root\_widget.strange\_bullet\_label.text=str(self.num\_strange\_bullets\_to\_fire)

'''

if root.main\_app.bMute\_SFX==False:

root.SFX\_strange\_bullets.play()

else:

if root.SFX\_strange\_bullets.state=='play':

root.SFX\_strange\_bullets.stop()'''

self.root\_widget.turret\_gun.add\_widget(bull)

return True

def update(self,dt):

# if hasattr(self, 'touch'):

# print(self.touch.pos)

if self.bshoot:

if self.touch.x-self.x==0:

self.ang= math.pi/2

else:

frac=0.0

self.ang=math.atan2(self.touch.y\*1.0-self.y , self.touch.x\*1.0-self.x)

if self.ang < math.pi\*frac and self.ang > -1\*math.pi/2:

self.ang=math.pi\*frac

elif self.ang > math.pi\*(1-frac) or self.ang < -1\*math.pi/2:

self.ang=math.pi\*(1-frac)

self.rotate\_instruction.angle= self.ang \* 180 / math.pi

return 0

class BallBase(Widget): #TODO : clean up atributtes

bmove=False

brotate=False

benemy=True

mass=0

radius=0

speed=0

rotation\_speed=0

rect=ObjectProperty(None)

def \_\_init\_\_ (self, root, ball\_size, ball\_position, density, \*\*kwargs):

self.ball\_size=ball\_size

self.ball\_position=ball\_position

self.density= density

self.speed=[0,0]

super(BallBase, self).\_\_init\_\_(\*\*kwargs)

self.update\_pos(root.main\_app.root)

def update\_pos (self, root):

diam=self.ball\_size\*root.width

self.radius=diam/2

self.mass=((diam/2)\*\*3) \* math.pi \* 4 / 3 \* self.density

self.size=(diam,diam)

self.center=root.width\* self.ball\_position[0],root.height\* self.ball\_position[1]

self.pos=[self.center[0]-self.radius,self.center[1]-self.radius]

def move(self, planet\_in\_list, root, dt):

self.pos[0]=self.pos[0]+dt\*self.speed[0]

self.pos[1]=self.pos[1]+dt\*self.speed[1]

planet\_in\_list[0][0]=self.pos[0]+self.radius

planet\_in\_list[0][1]=self.pos[1]+self.radius

self.rect.pos=self.pos

self.rot.origin=(self.pos[0] + self.radius, self.pos[1] + self.radius)

if (self.pos[1]>1.3 \* root.height) or (self.pos[1]< -0.3 \* root.height):

if isinstance(self, Objective\_Ball):

return 3

return 1

elif self.pos[0]<-0.2 \* root.width or self.pos[0]>root.width \* 1.2: # TODO: Physics!

if isinstance(self, Objective\_Ball):

return 3

return 1

factor\_for\_dvCM= 1.0/3

radiusmult= 2

ar=[0, 0]

for ball in root.balls\_list + root.balls\_objective\_list:

if ball[4] != self:

dy=-1\*ball[0][1]+self.pos[1] + self.radius

dx=-1\*ball[0][0]+self.pos[0] + self.radius

r2=dx\*\*2+dy\*\*2

r= math.sqrt(r2)

ang=math.atan2(dy,dx)

if (r <= radiusmult \* ( ball[1] + self.radius)): # whitin gravitation range

ball[4].bmove=True

if (r <= ( ball[1] + self.radius)): # planet collision

delta= math.sqrt( (-1\* r\*\*3 + 4 \* ((ball[2] \* 1.0 / ball[3] + self.mass \* 1.0 / self.density)/(4.0/3\*math.pi))) / (3 \* r) )

rbigger= 0.5 \* (r + delta)

rsmaller= r - rbigger

if (rsmaller < root.smallest\_poss\_planet\_radius):

rsmaller = 0.0

rbigger = ( (ball[2] \* 1.0 / ball[3] + self.mass \* 1.0 / self.density)/(4.0/3\*math.pi) ) \*\* (1.0/3)

if (self.radius >= ball[1]):

volbigger= rbigger\*\*3 \* 4.0/3 \* math.pi

massbigger= volbigger \* self.density

voltransfer = volbigger - self.mass \* 1.0 / self.density

masstransfer= voltransfer \* ball[3]

vang=math.atan2(ball[4].speed[1] , ball[4].speed[0])

dang=ang-vang

cdang=math.cos(dang)

dvCM=math.sqrt((ball[4].speed[1]\*\*2+ball[4].speed[0]\*\*2) \* 1.0 \* masstransfer \* (cdang \*\* 2) / self.mass) \* factor\_for\_dvCM

ratmass= math.sqrt(self.mass \* 1.0 /massbigger)

self.speed[0] = (self.speed[0] + dvCM \* math.cos(ang) ) \* ratmass

self.speed[1] = (self.speed[1] + dvCM \* math.sin(ang) ) \* ratmass

self.mass= massbigger

planet\_in\_list[2]=massbigger

ball[2]= (rsmaller \*\* 3) \* 4.0 / 3 \* math.pi \* ball[3]

ball[4].mass= ball[2]

biggersize= rbigger \* 1.0 / self.radius \* self.rect.size[0]

biggersize1= (biggersize, biggersize)

self.size= biggersize1

self.rect.size= biggersize1

smallersize= rsmaller \* 1.0 / ball[1] \* ball[4].rect.size[0]

smallersize1= (smallersize, smallersize)

ball[4].size= smallersize1

ball[4].rect.size= smallersize1

self.radius= rbigger

planet\_in\_list[1]= rbigger

ball[4].radius= rsmaller

ball[1]= rsmaller

else: # ball bigger than self

volbigger= rbigger\*\*3 \* 4.0/3 \* math.pi

massbigger= volbigger \* ball[3]

voltransfer = volbigger - ball[2] \* 1.0 / ball[3]

masstransfer= voltransfer \* self.density

vang=math.atan2(self.speed[1] , self.speed[0])

dang=vang - ang

cdang=math.cos(dang)

dvCM=math.sqrt((self.speed[1]\*\*2+self.speed[0]\*\*2) \* 1.0 \* masstransfer \* (cdang \*\* 2) / ball[2]) \* factor\_for\_dvCM

ratmass= math.sqrt(ball[2] \* 1.0 /massbigger)

ball[4].speed[0] = (ball[4].speed[0] - dvCM \* math.cos(ang) ) \* ratmass

ball[4].speed[1] = (ball[4].speed[1] - dvCM \* math.sin(ang) ) \* ratmass

ball[4].mass= massbigger

ball[2]= massbigger

self.mass= (rsmaller \*\* 3) \* 4.0 / 3 \* math.pi \* self.density

planet\_in\_list[2]=self.mass

smallersize= rsmaller \* 1.0 / self.radius \* self.rect.size[0]

smallersize1= (smallersize, smallersize)

self.size= smallersize1

self.rect.size= smallersize1

biggersize= rbigger \* 1.0 / ball[1] \* ball[4].rect.size[0]

biggersize1= (biggersize, biggersize)

ball[4].size= biggersize1

ball[4].rect.size= biggersize1

self.radius= rsmaller

planet\_in\_list[1]=rsmaller

ball[4].radius= rbigger

ball[1]= rbigger

# [ [ [center\_x, center\_y], radius, mass, density, widget\_ref ] , ...]

if ball[1]< root.smallest\_poss\_planet\_radius:

if ball in root.balls\_objective\_list:

root.remove\_widget(ball[4])

root.balls\_objective\_list.remove(ball)

root.num\_objective\_balls-=1

else:

root.remove\_widget(ball[4])

root.balls\_list.remove(ball)

else:

a = ( root.gravity\_constant \* ball[2] / float(r2) ) # TODO delete ball[3]

ar[0] = ar[0] - a \* math.cos(ang)

ar[1] = ar[1] - a \* math.sin(ang)

# [ [ [center\_x, center\_y], radius, mass, density, widget\_ref ] , ...]

if self.radius < root.smallest\_poss\_planet\_radius:

return 1

self.speed=[self.speed[0] + ar[0] \* dt , self.speed[1] + ar[1] \* dt ]

return 0

def colors\_and\_rotation(self):

pass

'''

with self.canvas.before:

PushMatrix()

Rotate (angle= random()\*360, origin= self.center) # TODO breaks the code

PopMatrix()

with self.canvas.before:

PushMatrix()

#Rotate (angle= random()\*360, origin= self.center) # TODO breaks the code

#Color(0.9375, 0.9375, 0.3671875)

with self.canvas.after:

PopMatrix()

'''

class Base\_Enemy(BallBase):

bbase\_enemy\_ball=True

def \_\_init\_\_ (self, root, ball\_size, ball\_position, density, \*\*kwargs):

super(Base\_Enemy, self).\_\_init\_\_(root, ball\_size, ball\_position, density, \*\*kwargs)

rand=random()

s=''

if root.main\_app.bArt\_White:

s='\_w'

else:

s=str(randint(1,root.main\_app.number\_of\_planets))

with self.canvas:

PushMatrix()

self.rot=Rotate (angle= rand\*360, origin= self.center) # TODO breaks the code

self.rect=Rectangle(source= 'Resources/balls/e' + s + '.png', pos=self.pos, size= self.size)

PopMatrix()

class Objective\_Ball(BallBase):

bbase\_objective\_ball=True

def \_\_init\_\_ (self, root, ball\_size, ball\_position, density, \*\*kwargs):

super(Objective\_Ball, self).\_\_init\_\_(root, ball\_size, ball\_position, density, \*\*kwargs)

rand=random()

s=''

if root.main\_app.bArt\_White:

s='\_w'

else:

s=str(randint(1,root.main\_app.number\_of\_planets))

with self.canvas:

PushMatrix()

self.rot=Rotate (angle= rand\*360, origin= self.center) # TODO breaks the code

self.rect=Rectangle(source= 'Resources/balls/o' + s + '.png', pos=self.pos, size= self.size)

PopMatrix()

class GameWidget(Widget):

main\_app=None

turret\_gun= ObjectProperty(None)

balls\_list=[] # [ [ [center\_x, center\_y], radius, mass, widget\_ref ] , ...]

# [i][0][0], [i][0][1], [i][1], [i][2], [i][3]

balls\_objective\_list=[] # [ [ [center\_x, center\_y], radius, mass, widget\_ref ] , ...]

gravity\_constant=50.0 / 4

bullet\_gravity\_constant= 1.0 \* 4

main\_planet= ObjectProperty(None)

current\_level=0

turret\_building= ObjectProperty(None)

num\_objective\_balls=0

heavy\_bullet\_constant=900.0

smallest\_poss\_planet\_radius= 0.75

SFX\_small\_Bullets=None

SFX\_Heavy\_Bullets=None

SFX\_strange\_bullets= None

sounds=None

count=0

count2=0

normal\_bullets\_label= None

heavy\_bullets\_label= None

heavy\_bullet\_graphic= None

strange\_bullet\_label= None

strange\_bullet\_graphic= None

def \_\_init\_\_(self,\*\*kwargs):

super(GameWidget, self).\_\_init\_\_(\*\*kwargs)

# ---- SFX sound

#self.SFX\_small\_Bullets = SoundSDL2(source= 'Resources/audio/bullet\_SFX.mp3')

#self.SFX\_Heavy\_Bullets = SoundSDL2(source= 'Resources/audio/Heavy\_Bullet\_SFX.mp3')

#self.SFX\_strange\_bullets= SoundSDL2(source= 'Resources/audio/strange\_bullet\_SFX.mp3')

#self.bind(size=self.update\_balls\_pos)

'''

def update\_balls\_pos(self, obj, value):

for child in self.children:

if child.bbase\_objective\_ball or child.bbase\_enemy\_ball:

child.update\_pos(self)

#[[child.center\_x,child.center\_y], child.radius, child.mass, 1.0, child]

for i in range(len(self.balls\_list)):

self.balls\_list[i]=[[self.balls\_list[i][4].center\_x, self.balls\_list[i][4].center\_y], self.balls\_list[i][4].radius, self.balls\_list[i][4].mass, self.balls\_list[i][4].density, self.balls\_list[i][4]]

for i in range(len(self.balls\_objective\_list)):

self.balls\_objective\_list[i]=[[self.balls\_objective\_list[i][4].center\_x, self.balls\_objective\_list[i][4].center\_y], self.balls\_objective\_list[i][4].radius, self.balls\_objective\_list[i][4].mass, self.balls\_objective\_list[i][4].density, self.balls\_objective\_list[i][4]]

'''

def update\_bullet\_graphic\_colors(self):

if self.turret\_gun.current\_ammo\_type==0:

with self.normal\_bullet\_graphic.canvas.before:

PushMatrix()

Color(0.9375, 0.9375, 0.3671875)

with self.normal\_bullet\_graphic.canvas.after:

PopMatrix()

if self.heavy\_bullet\_graphic != None:

self.heavy\_bullet\_graphic.canvas.before.clear()

self.heavy\_bullet\_graphic.canvas.after.clear()

if self.strange\_bullet\_graphic!= None:

self.strange\_bullet\_graphic.canvas.before.clear()

self.strange\_bullet\_graphic.canvas.after.clear()

elif self.turret\_gun.current\_ammo\_type==1 and self.heavy\_bullet\_graphic != None:

with self.heavy\_bullet\_graphic.canvas.before:

PushMatrix()

Color(0.9375, 0.9375, 0.3671875)

with self.heavy\_bullet\_graphic.canvas.after:

PopMatrix()

self.normal\_bullet\_graphic.canvas.before.clear()

self.normal\_bullet\_graphic.canvas.after.clear()

if self.strange\_bullet\_graphic!= None:

self.strange\_bullet\_graphic.canvas.before.clear()

self.strange\_bullet\_graphic.canvas.after.clear()

elif self.turret\_gun.current\_ammo\_type==2 and self.strange\_bullet\_graphic!= None:

with self.strange\_bullet\_graphic.canvas.before:

PushMatrix()

Color(0.9375, 0.9375, 0.3671875)

with self.strange\_bullet\_graphic.canvas.after:

PopMatrix()

self.normal\_bullet\_graphic.canvas.before.clear()

self.normal\_bullet\_graphic.canvas.after.clear()

if self.heavy\_bullet\_graphic != None:

self.heavy\_bullet\_graphic.canvas.before.clear()

self.heavy\_bullet\_graphic.canvas.after.clear()

def update(self, dt):

#print len(self.balls\_list)

'''

str1=""

for ball in self.balls\_list + self.balls\_objective\_list:

str1= str1 + "vx= " + str(ball[4].speed[0]) + " " + "vy= " + str(ball[4].speed[1]) + " ||| "

print str1

'''

for bullet in self.turret\_gun.children:

ret=bullet.update(self,dt)

if ret==1: # just remove bullet --------------------------------------------------------------------------------

self.turret\_gun.remove\_widget(bullet)

if isinstance(bullet, Bullet):

self.turret\_gun.num\_bullets\_be\_destroy-=1

elif isinstance(bullet, HeavyBullet):

self.turret\_gun.num\_heavy\_bullets\_to\_destroy-=1

elif isinstance(bullet, StrangeMatterBullet):

self.turret\_gun.num\_strange\_bullets\_to\_destroy-=1

#print "bullets to destroy " + str(self.turret\_gun.num\_bullets\_be\_destroy) + " bullets left " + str(self.turret\_gun.num\_bulletsy\_to\_fire)

elif ret==2: # hit objectve, objective ball removed in other function --------------------------------------------------------------------------------

self.turret\_gun.remove\_widget(bullet)

self.num\_objective\_balls-=1

if isinstance(bullet, Bullet):

self.turret\_gun.num\_bullets\_be\_destroy-=1

elif isinstance(bullet, StrangeMatterBullet):

self.turret\_gun.num\_strange\_bullets\_to\_destroy-=1

if self.num\_objective\_balls<=0:

self.main\_app.destroy\_level()

if self.current\_level == self.main\_app.beat\_to\_the\_level:

self.main\_app.beat\_to\_the\_level += 1

num=self.main\_app.beat\_to\_the\_level

with open('saves.txt','w') as filemap:

str1=""

if num < 10:

str1= "0"

str1+=str(num)

filemap.write(str1)

if self.current\_level==20:

self.main\_app.screen\_manager.current="level\_won\_screen\_end"

self.main\_app.dict\_space\_backgrounds['won\_end'].rotate()

else:

self.main\_app.screen\_manager.current="level\_won\_screen"

self.main\_app.dict\_space\_backgrounds['won'].rotate()

#print "Won the level"

def update\_balls(self,dt):

#dt=dt/4.0

'''

self.count+=dt

self.count2= (self.count2+1)%120

print(str(self.count) + " " + str(self.count2))

'''

bMovement=False

for planet in self.balls\_list:

if planet[4].bmove:

bMovement=True

ret=planet[4].move(planet, self, dt)

if ret==1: #remove planet

self.remove\_widget(planet[4])

self.balls\_list.remove(planet)

if planet[4].brotate:

pass

for planet in self.balls\_objective\_list:

if planet[4].bmove:

bMovement=True

ret=planet[4].move(planet, self, dt)

if ret==1: #remove planet

self.remove\_widget(planet[4])

self.balls\_objective\_list.remove(planet)

self.num\_objective\_balls-=1

if ret==3:

self.main\_app.dict\_space\_backgrounds['lost2'].rotate()

self.main\_app.screen\_manager.current="level\_lost\_screen\_2"

self.main\_app.destroy\_level()

if planet[4].brotate:

pass

if self.num\_objective\_balls<=0:

self.main\_app.destroy\_level()

if self.current\_level == self.main\_app.beat\_to\_the\_level:

self.main\_app.beat\_to\_the\_level += 1

num=self.main\_app.beat\_to\_the\_level

with open('saves.txt','w') as filemap:

str1=""

if num < 10:

str1= "0"

str1+=str(num)

filemap.write(str1)

if self.current\_level==20:

self.main\_app.screen\_manager.current="level\_won\_screen\_end"

self.main\_app.dict\_space\_backgrounds['won\_end'].rotate()

else:

self.main\_app.screen\_manager.current="level\_won\_screen"

self.main\_app.dict\_space\_backgrounds['won'].rotate()

elif self.turret\_gun.num\_bullets\_be\_destroy<=0 and self.turret\_gun.num\_heavy\_bullets\_to\_destroy<=0 and self.turret\_gun.num\_strange\_bullets\_to\_destroy<=0 and bMovement==False:

self.main\_app.dict\_space\_backgrounds['lost'].rotate()

self.main\_app.screen\_manager.current="level\_lost\_screen"

self.main\_app.destroy\_level()

def create\_ball (self, type, size\_frac, pos\_frac\_tuple, speed\_tuple=None):

if type=="base\_enemy":

base=Base\_Enemy(self, size\_frac, pos\_frac\_tuple, 1.0)

if speed\_tuple != None:

base.speed=[speed\_tuple[0] \* self.width, speed\_tuple[1] \* self.width]

base.bmove=True

self.add\_widget(base)

self.balls\_list.append([[base.center\_x, base.center\_y], base.radius, base.mass, base.density, base])

if type=="base\_objective":

base=Objective\_Ball(self, size\_frac, pos\_frac\_tuple, 1.0)

if speed\_tuple != None:

base.bmove=True

base.speed=[speed\_tuple[0] \* self.width, speed\_tuple[1] \* self.width]

self.add\_widget(base)

self.balls\_objective\_list.append([[base.center\_x, base.center\_y], base.radius, base.mass, base.density, base])

def schedule\_enemies(self):

# ( "enemy type", diameter= number\*width , (width\* number, heigth \* number) )

if self.current\_level==3:

self.create\_ball("base\_enemy", 0.22, (0.5, 0.5))

self.create\_ball("base\_enemy", 0.15, (0.2, 0.25))

self.create\_ball("base\_enemy", 0.15, (0.8, 0.25))

self.create\_ball("base\_enemy", 0.15, (0.2, 0.75))

self.create\_ball("base\_enemy", 0.15, (0.8, 0.75))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.17, (0.5, 0.9))

elif self.current\_level==1:

self.turret\_gun.create\_bullet\_count(20)

self.create\_ball("base\_enemy", 0.25, (0.2, 0.8))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.25, (0.8, 0.8))

elif self.current\_level==2:

self.turret\_gun.create\_bullet\_count(5)

self.create\_ball("base\_enemy", 0.2, (0.5, 0.65))

self.create\_ball("base\_enemy", 0.2, (0.5, 0.35))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.2, (0.5, 0.9))

elif self.current\_level==4:

self.turret\_gun.create\_bullet\_count(5)

self.create\_ball("base\_enemy", 0.2, (0.25, 0.71))

self.create\_ball("base\_enemy", 0.2, (1-0.25, 0.71))

#self.create\_ball("base\_enemy", 0.25, (0.5, 0.65))

self.create\_ball("base\_enemy", 0.3, (0.85, 0.4))

self.create\_ball("base\_enemy", 0.3, (0.15, 0.4))

self.num\_objective\_balls=2

self.create\_ball("base\_objective", 0.15, (0.15, 0.9))

self.create\_ball("base\_objective", 0.15, (0.85, 0.9))

elif self.current\_level==5:

self.turret\_gun.create\_bullet\_count(5)

self.create\_ball("base\_enemy", 0.23, (0.5, 0.35))

self.create\_ball("base\_enemy", 0.23, (0.5, 0.7))

self.num\_objective\_balls=2

self.create\_ball("base\_objective", 0.1, (0.5, 0.92))

self.create\_ball("base\_objective", 0.1, (0.5, 0.52))

elif self.current\_level==6:

self.turret\_gun.create\_bullet\_count(7)

self.create\_ball("base\_enemy", 0.23, (0.5, 0.7))

self.create\_ball("base\_enemy", 0.23, (0.15, 0.4))

self.create\_ball("base\_enemy", 0.23, (0.85, 0.4))

self.num\_objective\_balls=3

self.create\_ball("base\_objective", 0.15, (0.1, 0.65))

self.create\_ball("base\_objective", 0.15, (0.9, 0.65))

self.create\_ball("base\_objective", 0.15, (0.5, 0.92))

elif self.current\_level==7:

self.turret\_gun.create\_bullet\_count(7)

self.create\_ball("base\_enemy", 0.18, (0.5, 0.8))

self.create\_ball("base\_enemy", 0.18, (0.1, 0.8))

self.create\_ball("base\_enemy", 0.4, (1.0, 0.5))

self.num\_objective\_balls=2

self.create\_ball("base\_objective", 0.12, (0.3, 0.8))

self.create\_ball("base\_objective", 0.12, (0.7, 0.8))

elif self.current\_level==8:

self.turret\_gun.create\_bullet\_count(1,3)

self.create\_ball("base\_enemy", 0.2, (0.5, 0.7))

self.create\_ball("base\_enemy", 0.2, (0.9, 0.64))

self.create\_ball("base\_enemy", 0.2, (0.1, 0.64))

self.create\_ball("base\_enemy", 0.2, (0.3, 0.42))

self.create\_ball("base\_enemy", 0.2, (0.7, 0.42))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.15, (0.5, 0.9))

elif self.current\_level==9:

self.turret\_gun.create\_bullet\_count(3,2)

self.create\_ball("base\_enemy", 0.14, (-0.25, 0.5))

self.create\_ball("base\_enemy", 0.14, (0.05, 0.5))

self.create\_ball("base\_enemy", 0.14, (0.35, 0.5))

self.create\_ball("base\_enemy", 0.14, (0.65, 0.5))

self.create\_ball("base\_enemy", 0.14, (0.95, 0.5))

self.create\_ball("base\_enemy", 0.14, (1.25, 0.5))

self.create\_ball("base\_enemy", 0.14, (-0.1, 0.66))

self.create\_ball("base\_enemy", 0.14, (0.2, 0.66))

self.create\_ball("base\_enemy", 0.14, (0.5, 0.66))

self.create\_ball("base\_enemy", 0.14, (0.8, 0.66))

self.create\_ball("base\_enemy", 0.14, (1.1, 0.66))

self.create\_ball("base\_enemy", 0.14, (-0.1, 0.34))

self.create\_ball("base\_enemy", 0.14, (0.2, 0.34))

self.create\_ball("base\_enemy", 0.14, (0.5, 0.34))

self.create\_ball("base\_enemy", 0.14, (0.8, 0.34))

self.create\_ball("base\_enemy", 0.14, (1.1, 0.34))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.15, (0.5, 0.9))

elif self.current\_level==10:

self.turret\_gun.create\_bullet\_count(3,2)

self.create\_ball("base\_enemy", 0.15, (0.6, 0.7))

self.create\_ball("base\_enemy", 0.32, (1.22, 0.4))

self.create\_ball("base\_enemy", 0.24, (0.7, 0.4))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.14, (0.9, 0.8))

elif self.current\_level==11:

self.turret\_gun.create\_bullet\_count(3,2)

self.create\_ball("base\_enemy", 0.2, (0.19, 0.6))

self.create\_ball("base\_enemy", 0.2, (0.81, 0.6))

self.create\_ball("base\_enemy", 0.12, (0.5, 0.4))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.10, (0.5, 0.9))

elif self.current\_level==12:

self.turret\_gun.create\_bullet\_count(5)

self.create\_ball("base\_enemy", 0.15, (0.2, 0.75))

self.create\_ball("base\_enemy", 0.2, (0.8, 0.55))

self.create\_ball("base\_enemy", 0.2, (0.4, 0.35))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.10, (0.7, 0.95))

elif self.current\_level==13:

self.turret\_gun.create\_bullet\_count(0,2)

self.create\_ball("base\_enemy", 0.2, (0.1, 0.65))

self.create\_ball("base\_enemy", 0.2, (0.9, 0.65))

self.create\_ball("base\_enemy", 0.2, (0.27, 0.4))

self.create\_ball("base\_enemy", 0.2, (0.73, 0.4))

self.create\_ball("base\_enemy", 0.2, (0.5, 0.6))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.30, (0.5, 0.95))

elif self.current\_level==14:

self.num\_objective\_balls=4

self.turret\_gun.create\_bullet\_count(1,2)

self.create\_ball("base\_objective", 0.35, (0.5, 0.75))

self.create\_ball("base\_objective", 0.15, (0.9, 0.57))

self.create\_ball("base\_objective", 0.15, (0.13, 0.95))

self.create\_ball("base\_objective", 0.2, (0.35, 0.4))

elif self.current\_level==15:

self.turret\_gun.create\_bullet\_count(1,2)

self.create\_ball("base\_enemy", 0.2, (0.5, 0.4))

self.num\_objective\_balls=3

self.create\_ball("base\_objective", 0.15, (0.15, 0.9))

self.create\_ball("base\_objective", 0.15, (0.85, 0.9))

self.create\_ball("base\_objective", 0.2, (0.5, 0.7))

elif self.current\_level==16:

self.turret\_gun.create\_bullet\_count(3,0,2)

self.create\_ball("base\_enemy", 0.5, (0.0, 0.5))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.2, (0.7, 0.9))

elif self.current\_level==17:

self.turret\_gun.create\_bullet\_count(3,0,1)

self.create\_ball("base\_enemy", 0.2, (0.1, 0.7))

self.create\_ball("base\_enemy", 0.2, (0.9, 0.7))

self.create\_ball("base\_enemy", 0.35, (0.5, 0.5))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.20, (0.5, 0.9))

elif self.current\_level==18:

self.turret\_gun.create\_bullet\_count(3,0,1)

self.create\_ball("base\_enemy", 0.3, (0.7, 0.6))

self.create\_ball("base\_enemy", 0.4, (0.0, 0.4))

self.create\_ball("base\_enemy", 0.2, (0.2, 0.7))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.20, (0.5, 0.95))

elif self.current\_level==19:

self.turret\_gun.create\_bullet\_count(0,2,1)

self.create\_ball("base\_enemy", 0.17, (0.55, 0.35))

self.create\_ball("base\_enemy", 0.17, (0.55, 0.75))

self.num\_objective\_balls=2

self.create\_ball("base\_objective", 0.2, (0.35, 0.55))

self.create\_ball("base\_objective", 0.2, (0.75, 0.95))

elif self.current\_level==20:

self.turret\_gun.create\_bullet\_count(3,0,1)

self.create\_ball("base\_enemy", 0.17, (0.1, 0.4))

self.create\_ball("base\_enemy", 0.17, (0.9, 0.4))

self.create\_ball("base\_enemy", 0.3, (0.5, 0.5))

self.create\_ball("base\_enemy", 0.3, (0.5, 0.75))

self.num\_objective\_balls=1

self.create\_ball("base\_objective", 0.15, (0.5, 0.97))

elif self.current\_level==24:

self.turret\_gun.create\_bullet\_count(3,0,3)

self.create\_ball("base\_enemy", 0.1, (0.5, 0.3))

self.create\_ball("base\_enemy", 0.3, (0.5, 0.6))

self.create\_ball("base\_enemy", 0.2, (0.3, 0.9))

self.num\_objective\_balls=2

self.create\_ball("base\_objective", 0.2, (0.3, 0.7))

elif self.current\_level==25:

self.turret\_gun.create\_bullet\_count(3,3,3)

self.create\_ball("base\_enemy", 0.1, (0.5, 0.3))

self.create\_ball("base\_enemy", 0.3, (0.5, 0.6))

self.create\_ball("base\_enemy", 0.2, (0.3, 0.9))

self.num\_objective\_balls=2

self.create\_ball("base\_objective", 0.2, (0.3, 0.7))

class GameApp(App):

game\_widget= ObjectProperty(None)

screen\_manager= ObjectProperty(None)

# level\_selection\_gridlayout= ObjectProperty(None)

num\_buttons=20

bcreated\_first\_level=False

list\_clock\_repeat\_schedule=[0,0,0]

refresh\_rate=1.0/60

refresh\_rate\_balls\_physics= 1.0/60

beat\_to\_the\_level=0

bMute\_SFX=False # mute all sounds

bMute\_soundtrack=False

list\_soundtracks=[]

settings\_screen= None

number\_of\_backgrounds=6

number\_of\_planets=5

dict\_space\_backgrounds={}

bArt\_White= True

bFPS\_60T\_30F= True

conf=Config

def build(self):

self.icon= 'WEB\_icon.png'

self.title= 'Space Ballistics'

self.use\_kivy\_settings=False

background\_randint= randint(1,self.number\_of\_backgrounds)

background\_float0to1= random()

planet\_background\_randint=0

planet\_background\_float0to1=0

with open('saves.txt','r') as filemap:

filelines=filemap.readlines()

self.beat\_to\_the\_level= int(filelines[0][:2])

sm = ScreenManager(transition=FadeTransition())

self.screen\_manager=sm

start\_screen=StartScreen(name='start\_menu')

if self.beat\_to\_the\_level == 0:

start\_screen.game\_button.text='New Game'

self.button\_text=start\_screen.game\_button

planet\_background=Planet\_Background(self,planet\_background\_randint, planet\_background\_float0to1)

sm.bind(size=planet\_background.update\_pos)

start\_screen.add\_widget(planet\_background,10)

sm.add\_widget(start\_screen)

self.settings\_screen=SettingsScreen(name='settings')

planet\_background=Planet\_Background(self,planet\_background\_randint, planet\_background\_float0to1)

sm.bind(size=planet\_background.update\_pos)

self.settings\_screen.add\_widget(planet\_background, 10)

sm.add\_widget(self.settings\_screen)

#self.settings\_screen.checkbox\_Music.bind(active=self.checkbox.settings\_screen.)

level\_selection=LevelSelectionScreen(name='level\_selection')

planet\_background=Planet\_Background(self,planet\_background\_randint, planet\_background\_float0to1)

sm.bind(size=planet\_background.update\_pos)

level\_selection.add\_widget(planet\_background, 10)

# self.level\_selection\_gridlayout=level\_selection.gridlayout

self.button\_grid=level\_selection.gridlayout

for i in range(1,self.num\_buttons+1):

button=LevelSelectionButton()

button.level=i

button.text=str(i)

button.main\_app=self

level\_selection.gridlayout.add\_widget(button)

sm.add\_widget(level\_selection)

gamescreen=GameScreen(name='game\_screen')

space\_background=Space\_Background(self)

sm.bind(size=space\_background.update\_pos)

gamescreen.add\_widget(space\_background, 10)

gamescreen.space\_background=space\_background

self.screen\_game=gamescreen

game=GameWidget()

self.game\_widget=game

game.main\_app=self

gamescreen.add\_widget(game)

sm.add\_widget(gamescreen)

level\_won\_screen=LevelWonScreen(name='level\_won\_screen')

space\_background=Space\_Background(self, background\_randint, background\_float0to1)

sm.bind(size=space\_background.update\_pos)

level\_won\_screen.add\_widget(space\_background, 10)

sm.add\_widget(level\_won\_screen)

self.dict\_space\_backgrounds['won']=space\_background

level\_won\_screen\_end=LevelWonScreenEnd(name='level\_won\_screen\_end')

space\_background=Space\_Background(self, background\_randint, background\_float0to1)

sm.bind(size=space\_background.update\_pos)

level\_won\_screen\_end.add\_widget(space\_background, 10)

sm.add\_widget(level\_won\_screen\_end)

self.dict\_space\_backgrounds['won\_end']=space\_background

level\_lost\_screen=LevelLostScreen(name='level\_lost\_screen')

space\_background=Space\_Background(self, background\_randint, background\_float0to1)

sm.bind(size=space\_background.update\_pos)

level\_lost\_screen.add\_widget(space\_background, 10)

sm.add\_widget(level\_lost\_screen)

self.dict\_space\_backgrounds['lost']=space\_background

lost2=LevelLostScreen2(name='level\_lost\_screen\_2')

lost2.lost\_label.text="Level Lost:\nPlanet out of range"

space\_background=Space\_Background(self, background\_randint, background\_float0to1)

sm.bind(size=space\_background.update\_pos)

lost2.add\_widget(space\_background, 10)

self.dict\_space\_backgrounds['lost2']=space\_background

sm.add\_widget(lost2)

EventLoop.window.bind(on\_keyboard=self.hook\_keyboard)

#sm.current = "game\_screen"

# TODO

game.screen\_manager=sm

if self.beat\_to\_the\_level==0:

self.beat\_to\_the\_level=1

with open('saves.txt','w') as filemap:

filemap.write('01')

self.update\_button\_colors()

# audio soundtrack

#self.list\_soundtracks.append(SoundSDL2(source= 'Resources/audio/chimes\_soundtrack.mp3'))

#self.list\_soundtracks.append(SoundSDL2(source= 'Resources/audio/Chimera.mp3'))

#self.soundtrack\_position= randint(0,len(self.list\_soundtracks)-1)

#Clock.schedule\_interval(self.update\_soundtrack\_music, 1.0 / 2)

#self.update\_soundtrack\_music()

return sm

def on\_pause(self):

return True

#def update\_soundtrack\_music (self):

# if self.bMute\_soundtrack==False:

# if self.list\_soundtracks[self.soundtrack\_position].state!="play":

# self.soundtrack\_position= (self.soundtrack\_position+1) % len(self.list\_soundtracks)

# self.list\_soundtracks[self.soundtrack\_position].play()

# else:

# if self.list\_soundtracks[self.soundtrack\_position].state=="play":

# self.list\_soundtracks[self.soundtrack\_position].stop()

#self.soundtrack\_position= (self.soundtrack\_position+1) % len(self.list\_soundtracks)

def update\_button\_colors(self):

for button in self.button\_grid.children:

if button.level <= self.beat\_to\_the\_level:

button.disabled=False

else:

button.disabled=True

'''

if button.level <= self.beat\_to\_the\_level:

button.background\_color= (0,1,0,1)

else:

button.background\_color= (1,0,0,1)

'''

def destroy\_level(self):

if self.bcreated\_first\_level==True:

#self.game\_widget.current\_level=0

for event in self.list\_clock\_repeat\_schedule:

Clock.unschedule(event)

self.game\_widget.balls\_list=[]

self.game\_widget.balls\_objective\_list=[]

self.game\_widget.clear\_widgets()

def create\_planet\_and\_turret(self, type):

if type == "flat":

planet=Planet()

'''

planet.pos= 0, 0

planet.size= self.root.width,self.root.width \* 0.212037037

with planet.canvas:

Rectangle(source= 'Resources/Spaceship.png', pos= planet.pos, size= planet.size)

'''

planet.pos= 0, 0 #-self.root.width/ 20.0

planet.size= self.root.width,self.root.width \* 2.5 / 9

planet.radius= planet.width/2

planet.mass= (planet.width\*\*3.0) / 8 \* 3.141593 \* 4 / 3

s=''

if self.bArt\_White:

s='\_w'

with planet.canvas:

Rectangle(source= 'Resources/planet'+s+'.png', pos= planet.pos, size= planet.size)

self.game\_widget.main\_planet=planet

turret\_building=TurretBuilding()

turret\_building.pos= planet.center\_x-planet.width/7, planet.width/5.4

turret\_building.size=0.3\*planet.width,0.3\*planet.width

s=''

if self.bArt\_White:

s='\_w'

with turret\_building.canvas:

Rectangle(source= 'Resources/turret\_building'+s+'.png', pos= turret\_building.pos, size= turret\_building.size)

turret\_gun=TurretGun()

turret\_gun.root\_widget= self.game\_widget

turret\_gun.pos= turret\_building.x + turret\_building.width/2, turret\_building.y + turret\_building.height\*2/7.0

turret\_gun.size=0.1\*planet.width , 0.25 \* (0.1 \* planet.width)

turret\_barrel= TurretBarrel()

s=''

if self.bArt\_White:

s='\_w'

with turret\_barrel.canvas:

Rectangle(source= 'Resources/barrel'+s+'.png', pos= turret\_gun.pos, size= turret\_gun.size)

with turret\_barrel.canvas.before:

PushMatrix()

turret\_gun.rotate\_instruction= Rotate (angle= 0, origin= (turret\_gun.pos[0], turret\_gun.pos[1]+ 0.25 \* (0.1 \* planet.width) / 2))

with turret\_barrel.canvas.after:

PopMatrix()

self.game\_widget.turret\_building=turret\_building

self.game\_widget.turret\_gun=turret\_gun

turret\_gun.barrel\_length = turret\_gun.root\_widget.turret\_building.width/3.2

self.game\_widget.add\_widget(turret\_gun)

self.game\_widget.add\_widget(turret\_barrel)

self.game\_widget.add\_widget(turret\_building)

self.game\_widget.add\_widget(planet)

'''

elif type == "round old":

planet=Planet()

planet.pos= self.root.width/2-self.root.width\*0.75/2,-self.root.width\*0.6

planet.size= self.root.width\*0.75,self.root.width\*0.75

planet.radius= planet.width/2

planet.mass= planet.width\*planet.width/4\*3.141593

with planet.canvas:

Rectangle(source= 'Resources/planet\_full.png', pos= planet.pos, size= planet.size)

self.game\_widget.main\_planet=planet

turret\_building=TurretBuilding()

turret\_building.pos= planet.center\_x-planet.width/7, planet.center\_y+planet.width/2-planet.width/70

turret\_building.size=0.3\*planet.width,0.3\*planet.width

with turret\_building.canvas:

Rectangle(source= 'Resources/turret\_building.png', pos= turret\_building.pos, size= turret\_building.size)

turret\_gun=TurretGun()

turret\_gun.root\_widget= self.game\_widget

turret\_gun.pos= turret\_building.x + turret\_building.width/2, turret\_building.y + turret\_building.height\*2/5

turret\_gun.barrel\_length = turret\_gun.root\_widget.turret\_building.width/3.0

self.game\_widget.turret\_gun=turret\_gun

self.game\_widget.add\_widget(turret\_gun)

self.game\_widget.add\_widget(turret\_building)

self.game\_widget.add\_widget(planet)

'''

def create\_additional\_widgets\_on\_level\_creation (self):

butt\_size = self.screen\_manager.width/14.0

factor= 1.2

replay\_butt= Button()

replay\_butt.background\_normal= 'Resources/restart\_button.png'

replay\_butt.background\_down= 'Resources/restart\_button.png'

replay\_butt.size=(butt\_size, butt\_size)

replay\_butt.pos= (self.screen\_manager.width - factor \* butt\_size, self.screen\_manager.height - factor \* butt\_size)

replay\_butt.border= (0,0,0,0)

self.game\_widget.add\_widget(replay\_butt)

replay\_butt.bind(on\_press=self.replay\_button\_f)

music\_butt= ToggleButton()

music\_butt.background\_normal= 'Resources/Music\_button.png'

music\_butt.background\_down= 'Resources/Music\_button\_off.png'

# if (self.bMute\_SFX and self.bMute\_soundtrack):

# music\_butt.state='down'

music\_butt.size=(butt\_size, butt\_size)

music\_butt.pos= (self.screen\_manager.width - factor \* 2 \* butt\_size, self.screen\_manager.height - factor \* butt\_size)

music\_butt.border= (0,0,0,0)

self.game\_widget.add\_widget(music\_butt)

music\_butt.bind(on\_press=self.music\_button\_f)

normal\_bullets\_label= Label()

self.game\_widget.normal\_bullets\_label=normal\_bullets\_label

normal\_bullets\_label.text= str(self.game\_widget.turret\_gun.num\_bulletsy\_to\_fire)

normal\_bullets\_label.pos= (0.4 \* butt\_size, self.screen\_manager.height - 1.0 \* butt\_size)

normal\_bullets\_label.size=(butt\_size, butt\_size )

normal\_bullets\_label.text\_size= (butt\_size, butt\_size )

normal\_bullets\_label.halign= 'right'

#normal\_bullets\_label.shorten\_from="left"

#normal\_bullets\_label.shorten=True

self.game\_widget.add\_widget(normal\_bullets\_label)

normal\_bullet\_graphic= Widget()

self.game\_widget.normal\_bullet\_graphic= normal\_bullet\_graphic

normal\_bullet\_graphic.pos= (normal\_bullets\_label.x + butt\_size/0.9 , normal\_bullets\_label.y - butt\_size / 8.0)

normal\_bullet\_graphic.size= (butt\_size/1.2, butt\_size/1.2)

with normal\_bullet\_graphic.canvas:

Rectangle(source="Resources/normal\_bullets\_graphic.png", pos= normal\_bullet\_graphic.pos, size= normal\_bullet\_graphic.size)

self.game\_widget.add\_widget(normal\_bullet\_graphic)

diff=0

if self.game\_widget.turret\_gun.num\_heavy\_bullets\_to\_fire !=0:

heavy\_bullets\_label= Label()

self.game\_widget.heavy\_bullets\_label=heavy\_bullets\_label

heavy\_bullets\_label.text= str(self.game\_widget.turret\_gun.num\_heavy\_bullets\_to\_fire)

heavy\_bullets\_label.pos= (2.5 \* butt\_size, normal\_bullets\_label.y)

heavy\_bullets\_label.size=(butt\_size, butt\_size )

heavy\_bullets\_label.text\_size= (butt\_size, butt\_size )

heavy\_bullets\_label.halign= 'right'

#heavy\_bullets\_label.shorten\_from="left"

#heavy\_bullets\_label.shorten=True

self.game\_widget.add\_widget(heavy\_bullets\_label)

heavy\_bullet\_graphic= Widget()

self.game\_widget.heavy\_bullet\_graphic= heavy\_bullet\_graphic

heavy\_bullet\_graphic.pos= (heavy\_bullets\_label.x + butt\_size/0.9 , heavy\_bullets\_label.y - butt\_size / 8.0)

heavy\_bullet\_graphic.size= (butt\_size/1.2, butt\_size/1.2)

with heavy\_bullet\_graphic.canvas:

Rectangle(source="Resources/heavy\_bullets\_graphic.png", pos= heavy\_bullet\_graphic.pos, size= heavy\_bullet\_graphic.size)

self.game\_widget.add\_widget(heavy\_bullet\_graphic)

diff=butt\_size \* 2.0

if self.game\_widget.turret\_gun.num\_strange\_bullets\_to\_fire !=0:

strange\_bullet\_label= Label()

self.game\_widget.strange\_bullet\_label=strange\_bullet\_label

strange\_bullet\_label.text= str(self.game\_widget.turret\_gun.num\_strange\_bullets\_to\_fire)

strange\_bullet\_label.pos= (2.5 \* butt\_size + diff, normal\_bullets\_label.y)

strange\_bullet\_label.size=(butt\_size, butt\_size )

strange\_bullet\_label.text\_size= (butt\_size, butt\_size )

strange\_bullet\_label.halign= 'right'

#strange\_bullet\_label.shorten\_from="left"

#strange\_bullet\_label.shorten=True

self.game\_widget.add\_widget(strange\_bullet\_label)

strange\_bullet\_graphic= Widget()

self.game\_widget.strange\_bullet\_graphic= strange\_bullet\_graphic

strange\_bullet\_graphic.pos= (strange\_bullet\_label.x + butt\_size/0.9 , strange\_bullet\_label.y - butt\_size / 8.0)

strange\_bullet\_graphic.size= (butt\_size/1.2, butt\_size/1.2)

with strange\_bullet\_graphic.canvas:

Rectangle(source="Resources/strange\_bullets\_graphic.png", pos= strange\_bullet\_graphic.pos, size= strange\_bullet\_graphic.size)

self.game\_widget.add\_widget(strange\_bullet\_graphic)

self.game\_widget.update\_bullet\_graphic\_colors()

def create\_level(self):

'''

if self.bFPS\_60T\_30F:

Config.set('graphics', 'maxfps', 60)

Config.write()

else:

Config.set('graphics', 'maxfps', 30)

Config.write()

'''

if (self.game\_widget.current\_level>=1) and (self.game\_widget.current\_level<=24):

self.create\_planet\_and\_turret('flat')

elif (self.game\_widget.current\_level==25):

self.create\_planet\_and\_turret('flat')

if (self.game\_widget.current\_level==8) and self.beat\_to\_the\_level==8:

label=Label(text='Double tap anywhere on the main planet (below the turret) to change ammo type.')

popup = Popup(title='Heavy Bullets available', content=label ,size\_hint=(0.7, 0.4))

popup.title\_align='center'

label.text\_size=(self.screen\_manager.width \* 0.5 , None)

popup.title\_size= '20sp'

popup.open()

elif (self.game\_widget.current\_level==16) and self.beat\_to\_the\_level==16:

label=Label(text='Double tap anywhere on the main planet (below the turret) to change ammo type.')

popup = Popup(title='Strange Matter Bullets available', content=label ,size\_hint=(0.7, 0.4))

popup.title\_align='center'

label.text\_size=(self.screen\_manager.width \* 0.5 , None)

popup.title\_size= '20sp'

popup.open()

self.game\_widget.schedule\_enemies()

self.create\_additional\_widgets\_on\_level\_creation()

self.list\_clock\_repeat\_schedule[0]=Clock.schedule\_interval(self.game\_widget.update, self.refresh\_rate)

self.list\_clock\_repeat\_schedule[1]=Clock.schedule\_interval(self.game\_widget.turret\_gun.update, self.refresh\_rate)

self.list\_clock\_repeat\_schedule[2]=Clock.schedule\_interval(self.game\_widget.update\_balls, self.refresh\_rate\_balls\_physics)

self.bcreated\_first\_level=True

def hook\_keyboard(self, window, key, \*largs):

if key == 27:

if self.screen\_manager.current == "start\_menu":

self.get\_running\_app().stop()

elif self.screen\_manager.current == "settings":

self.screen\_manager.current = "start\_menu"

elif self.screen\_manager.current == "level\_selection":

if self.beat\_to\_the\_level > 0:

self.button\_text.text='Continue'

self.screen\_manager.current = "start\_menu"

elif self.screen\_manager.current == "level\_lost\_screen":

self.update\_button\_colors()

self.screen\_manager.current = "level\_selection"

elif self.screen\_manager.current == "level\_lost\_screen\_2":

self.update\_button\_colors()

self.screen\_manager.current = "level\_selection"

elif self.screen\_manager.current == "level\_won\_screen":

self.update\_button\_colors()

self.screen\_manager.current = "level\_selection"

elif self.screen\_manager.current == "game\_screen":

self.destroy\_level()

self.update\_button\_colors()

self.screen\_manager.current = "level\_selection"

elif self.screen\_manager.current == "level\_won\_screen\_end":

self.update\_button\_colors()

self.screen\_manager.current = "level\_selection"

return True

def replay\_button\_f (self, button):

self.destroy\_level()

self.create\_level()

def music\_button\_f (self, toggle\_button):

return

''' if toggle\_button.state == 'normal':

self.bMute\_SFX= False

self.bMute\_soundtrack= False

self.settings\_screen.checkbox\_Music.active=False

self.settings\_screen.checkbox\_SFX.active=False

else:

self.bMute\_SFX= True

self.bMute\_soundtrack= True

self.settings\_screen.checkbox\_Music.active=True

self.settings\_screen.checkbox\_SFX.active=True'''

# self.update\_soundtrack\_music()

if \_\_name\_\_ == '\_\_main\_\_':

GameApp().run()

#:kivy 1.9.0

<Button>:

background\_normal: 'Resources/button\_normal.png'

<Space\_Background>:

canvas:

Rectangle:

source: 'Resources/backgrounds/space\_bk.png'

pos: 0,0

size: app.screen\_manager.height , app.screen\_manager.height

<HeavyBullet>:

canvas:

Ellipse:

pos: self.x - app.root.width/80, self.y - app.root.width/80

size: app.root.width/40 , app.root.width/40

<StrangeMatterBullet>:

canvas:

Rectangle:

source: 'Resources/strange\_bullet\_bullet.png'

pos: self.x - app.root.width/80, self.y - app.root.width/80

size: app.root.width/40 , app.root.width/40

<Bullet>:

canvas:

Ellipse:

pos: self.x - app.root.width/160, self.y - app.root.width/160

size: app.root.width/80 , app.root.width/80

<SettingsScreen>:

checkbox\_SFX: sfxmuteid

checkbox\_Music: musicid

Label:

text: "[b]Settings[/b]"

pos\_hint: {'y': 0.35}

font\_name: 'moonhouse'

font\_size: '50sp'

markup: True

halign: 'center'

Label:

text: "The Soundtracks in this game were modified from music found at http://www.purple-planet.com"

font\_size: '9sp'

pos\_hint: {'y': -0.47}

markup: True

halign: 'center'

GridLayout:

spacing: root.width/40

padding: root.width/20

size\_hint: 0.9, 0.3

pos\_hint: {'y': 0.45}

cols: 2

Label:

text: "Mute sound effects"

CheckBox:

id: sfxmuteid

size\_hint\_x: 0.2

active: False

on\_active:

app.bMute\_SFX=sfxmuteid.active

Label:

text: "Mute Music"

CheckBox:

id: musicid

size\_hint\_x: 0.2

active: False

on\_active:

app.bMute\_soundtrack=musicid.active

#app.update\_soundtrack\_music()

Label:

text: "Choose Level art style:"

pos\_hint: {'y': -0.06}

GridLayout:

spacing: root.width/40

padding: root.width/20

size\_hint: 0.8, 0.3

pos\_hint: {'x': 0.1, 'y': 0.2}

cols: 4

Label:

text: "simple (default)"

CheckBox:

id: art1id

group: 'art'

size\_hint\_x: 0.2

active: True

on\_active:

app.bArt\_White=art1id.active

Label:

text: "textured"

CheckBox:

id: art2id

group: 'art'

size\_hint\_x: 0.2

active: False

on\_active:

app.bArt\_White= not (art2id.active)

Label:

text: "Choose MAX FPS (set at 30 if the application is lagging):"

pos\_hint: {'y': -0.27}

GridLayout:

spacing: root.width/40

padding: root.width/20

size\_hint: 0.8, 0.3

pos\_hint: {'x': 0.1, 'y': 0.0}

cols: 4

Label:

text: "30"

CheckBox:

id: fps1id

group: 'fps'

size\_hint\_x: 0.2

active: False

on\_active:

app.bFPS\_60T\_30F= not(fps1id.active)

app.refresh\_rate= 1.0/30

app.refresh\_rate\_balls\_physics= 1.0/30

app.conf.set('graphics', 'maxfps', 30)

app.conf.write()

Label:

text: "60 (default)"

CheckBox:

id: fps2id

group: 'fps'

size\_hint\_x: 0.2

active: True

on\_active:

app.bFPS\_60T\_30F= fps2id.active

app.refresh\_rate= 1.0/60

app.refresh\_rate\_balls\_physics= 1.0/60

app.conf.set('graphics', 'maxfps', 60)

app.conf.write()

<LevelSelectionButton>:

background\_normal: 'Resources/button\_normal.png'

background\_disabled\_normal: 'Resources/button\_disabled\_normal.png'

size\_hint: 0.2, 0.2

<LevelLostScreen>:

lost\_label: textid

FloatLayout:

Label:

id: textid

text: "Level Lost:\nRan out of bullets"

font\_size: '40sp'

markup: True

halign: 'center'

pos\_hint: {'y': 0.3}

BoxLayout:

orientation: 'vertical'

padding: root.width/8

spacing: root.width/12.0

size\_hint: 1, 0.5

pos\_hint: {'y': 0.12}

Button:

text: 'Try Again'

on\_press:

app.create\_level()

app.screen\_manager.current = 'game\_screen'

size\_hint: 1, 0.15

Button:

text: 'Main Menu'

on\_press:

app.button\_text.text='Continue'

root.manager.current = 'start\_menu'

size\_hint: 1, 0.15

<LevelWonScreenEnd>:

FloatLayout:

Label:

text: "[b]Level Won[/b][size=25]\n\nThanks for Playing![/size]"

font\_size: '50sp'

markup: True

halign: 'center'

pos\_hint: {'y': 0.3}

BoxLayout:

orientation: 'vertical'

padding: root.width/8.0

spacing: root.width/12.0

size\_hint: 1, 0.5

pos\_hint: {'y': 0.07}

Button:

text: 'Try Again'

on\_press:

app.create\_level()

app.screen\_manager.current = 'game\_screen'

size\_hint: 1, 0.15

Button:

text: 'Main Menu'

on\_press:

app.button\_text.text='Continue'

root.manager.current = 'start\_menu'

size\_hint: 1, 0.15

<LevelWonScreen>:

FloatLayout:

Label:

text: "[b]Level Won[/b]"

font\_size: '50sp'

markup: True

halign: 'center'

pos\_hint: {'y': 0.35}

BoxLayout:

orientation: 'vertical'

padding: root.width/8.0

spacing: root.width/12.0

size\_hint: 1, 0.65

pos\_hint: {'y': 0.07}

Button:

text: 'Next Level'

on\_press:

app.game\_widget.current\_level+=1

app.create\_level()

app.screen\_game.space\_background.rotate()

app.screen\_manager.current = 'game\_screen'

size\_hint: 1, 0.15

Button:

text: 'Try Again'

on\_press:

app.create\_level()

app.screen\_manager.current = 'game\_screen'

size\_hint: 1, 0.15

Button:

text: 'Main Menu'

on\_press:

app.button\_text.text='Continue'

root.manager.current = 'start\_menu'

size\_hint: 1, 0.15

<StartScreen>:

game\_button: gamebuttonid

FloatLayout:

Label:

font\_name: 'moonhouse'

text: "[size=80]S[/size]pace\n\n [size=80]B[/size]allistics"

font\_size: '50sp'

markup: True

halign: 'left'

pos\_hint: {'y': 0.33}

BoxLayout:

orientation: 'vertical'

padding: root.width/8.0

spacing: root.width/20

size\_hint: 1, 0.6

pos\_hint: {'y': 0.04}

Button:

id: gamebuttonid

text: 'Continue'

on\_press: root.manager.current = 'level\_selection'

size\_hint: 1, 0.12

Button:

text: 'Settings'

on\_press: root.manager.current = 'settings'

size\_hint: 1, 0.12

Button:

text: 'Quit'

on\_press: app.stop()

size\_hint: 1, 0.12

<LevelSelectionScreen>:

gridlayout: gridlayoutid

Label:

text: "Select Level"

pos\_hint: {'y': 0.3}

font\_name: 'moonhouse'

font\_size: '50sp'

markup: True

halign: 'center'

GridLayout:

id: gridlayoutid

cols: 5

rows: 4

spacing: root.width/40

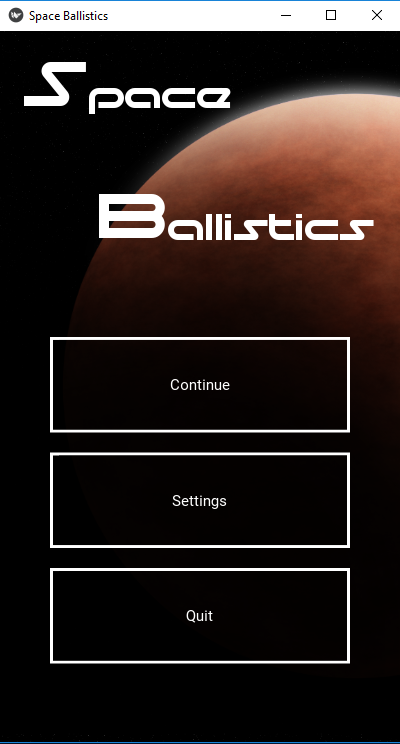
padding: root.width/10.0

size\_hint: 1, 0.50

pos\_hint: {'y': 0.165}

* **Dokumentasi :**

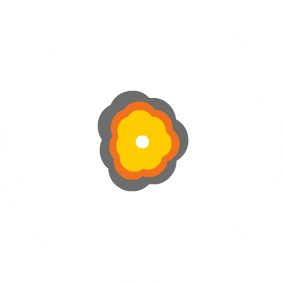
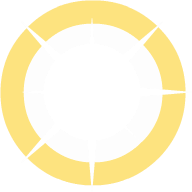
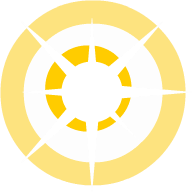
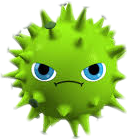
****

****

**DOKUMENTASI ASET PYGAME DAN KIVY**

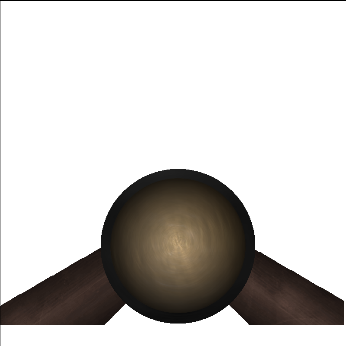
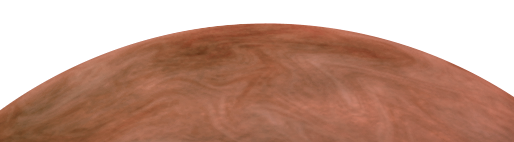
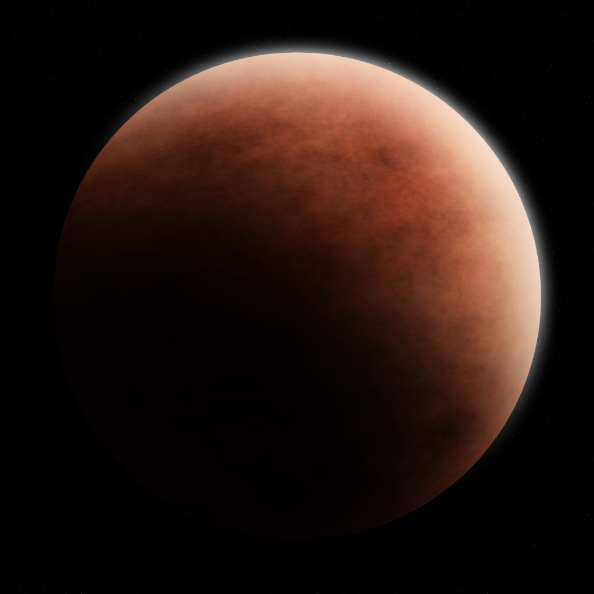
**PEMROGRAMAN BERORIENTASI**

**PYGAME**

****

****

**KIVY**

****

**PENUTUP**

**Kegunaan Laporan**

Laporan ini dibuat sebagai salah satu syarat mengikuti Ujian Akhir semester (UAS),dan sebagai pedoman atau tuntunan utuk kita belajar.

**Kesimpulan.**

Setelah saya menyelesaikan tugas akhir ini, kami dapat mengetahui bahwa betapa pentingnya laporan tugas akhir ini,karna loparan tugas akhir ini kami dapat mengikuti Ujian Akhir Semster Mata Kulia Pemrograman Berorientasi Objek (PBO).

**DAFTAR PUSTAKA**

**kivy=https://github.com/Ross65536/kivy-ballistics**

**pygame=https://github.com/tasdikrahman/spaceShooter**