**DOKUMENTASI TUGAS PYGAME & KIVY**

**PEMROGRAMAN BERORIENTASI OBJECT**



**Nama Kelompok :**

1. Bagus Fajariyanto (170411100045)
2. Musfirotummamlu’ah (170411100051)

**TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**Pygame**

1. **Definisi Pygame**

Pygame merupakan modul python yang dapat digunakan untuk pembuatan Game. Agar modul pygame dapat digunakan untuk pembuatan game, maka terlebih dahulu dilakukan instalasi Pygame.

1. **Instalasi Pygame**
2. Siapkan file pygame.whl

Download library pygame di https://www.lfd.uci.edu/~gohlke/pythonlibs/ .Sesuaikan versi python yang terinstall dengan versi library pygame yang akan di download. Contoh python v3.5 telah terinstall, maka download library pygame dengan cp35m-win32.

1. Rubah file pygame.whl menjadi .zip

Rubah extension file library pygame lalu extract isi library tersebut.

1. Copy file pygame yang dibutuhkan
2. masuk ke dalam directory python (C:\Users\#username\AppData\Local\Programs\Python\Python35-32)
3. masuk kedalam folder “include” dan buat folder baru bernama “pygame”.
4. Di dalam folder hasil extract file library pygame yang sudah didownload, masuk ke “pygame1.9.4.data\header”, copy semua file di dalam folder tersebut dan masukkan ke dalam folder: C:\Users\#username\AppData\Local\Programs\Python\Python35-32\include\pygame
5. kembali ke folder hasil extract file library pygame tadi, copy folder “pygame” dan “pygame1.9.4.dist-info” kedalam: C:\Users\#username\AppData\Local\Programs\Python\Python35-32\Lib\site-packages
6. Cek hasil installasi Buka IDLE Python lalu lakukan perintah “import pygame”, jika tidak ada tulisan error maka pygame sudah berhasil terinstall.

1. **Penjelasan OOP**

Untuk dapat menggunakan method-method ataupun event yang terdapat pada modul pygame, maka terlebih dahulu dilakukan perintah import pygame, seperti contoh berikut ini :

import pygame

**Class, property dan method :**

Pygame.sprite.Sprite

|  |
| --- |
| Wall |
| * Image * rect |
|  |

|  |
| --- |
| bigBlock |
| * meteor * image * rect |
| + update()  + reset\_pos() |

|  |
| --- |
| Player |
| * rocket * image * rect * change * walls * width * height |
| + update()  + changespeed() |

|  |
| --- |
| Bullet |
| * Image * rect * floating\_point * change |
| + update() |

|  |
| --- |
| Block |
| * meteor * image * rect |
| + update()  + reset\_pos() |

Konsep OOP :

Class Wall, Block, bigBlock,Player dan Bullet merupakan inheritance dari class bawaan dari pygame yaitu Pygame.sprite.Sprite

**Public Variable :**

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

BLUE = (0, 0, 255)

GREEN = (0, 255, 0)

PURPLE = (255, 0, 255)

SCREEN\_WIDTH = 500

SCREEN\_HEIGHT = 650

Gambar = pygame.image.load('Background.png')

Background = pygame.transform.smoothscale(Gambar,(SCREEN\_WIDTH,SCREEN\_HEIGHT))

**Method and Class from pygame :**

+ init()

+ sprite.Sprite

+ surface.Surface

+ sprite.Group

+ time.Clock

+ display.set\_mode

+ event.get

+ event.type

+ event.key

+ sprite.Group.draw

+ display.flip

+ time.Clock.tick

+ quit()

1. Class Wall

Memiliki property : image dan rect

1. Block

Memiliki property : image, rect dan meteor

Memiliki method :

1. reset\_pos : method yang mengatur saat meteor kecil jatuh di screen
2. update : method yang akan otomatis terpanggil ketika meteor kecil sudah habis dan akan kembali ke ats screen
3. bigBlock

Memiliki property : image, rect dan meteor

Memiliki method :

1. reset\_pos : method yang mengatur saat meteor besar jatuh di screen
2. update : method yang akan otomatis terpanggil ketika meteor besar sudah habis dan akan kembali ke ats screen
3. Player

Memiliki property : image, rect, rocket, change, walls, width, height

Memiliki method :

1. changespeed : mengubah kecepatan pesawat
2. update: mengatur pesawat bergerak ke kanan atau ke kiri
3. Bullet

Memiliki property : image, rect, floating\_point dan change

Memiliki method :

1. update : mengatur keluarnya peluru pesawat sesuai dengan posisi kursor
2. **Program**
3. Sebelum modifikasi

"""

Show how to fire bullets at the mouse.

Sample Python/Pygame Programs

Simpson College Computer Science

http://programarcadegames.com/

http://simpson.edu/computer-science/

"""

import pygame

import random

import math

# Define some colors

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

BLUE = (0, 0, 255)

SCREEN\_WIDTH = 700

SCREEN\_HEIGHT = 400

# --- Classes

class Block(pygame.sprite.Sprite):

""" This class represents the block. """

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

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

self.image = pygame.Surface([20, 15])

self.image.fill(color)

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

class Player(pygame.sprite.Sprite):

""" This class represents the Player. """

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

""" Set up the player on creation. """

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

self.image = pygame.Surface([20, 20])

self.image.fill(RED)

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

class Bullet(pygame.sprite.Sprite):

""" This class represents the bullet. """

def \_\_init\_\_(self, start\_x, start\_y, dest\_x, dest\_y):

""" Constructor.

It takes in the starting x and y location.

It also takes in the destination x and y position.

"""

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

# Set up the image for the bullet

self.image = pygame.Surface([4, 10])

self.image.fill(BLACK)

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

# Move the bullet to our starting location

self.rect.x = start\_x

self.rect.y = start\_y

# Because rect.x and rect.y are automatically converted

# to integers, we need to create different variables that

# store the location as floating point numbers. Integers

# are not accurate enough for aiming.

self.floating\_point\_x = start\_x

self.floating\_point\_y = start\_y

# Calculation the angle in radians between the start points

# and end points. This is the angle the bullet will travel.

x\_diff = dest\_x - start\_x

y\_diff = dest\_y - start\_y

angle = math.atan2(y\_diff, x\_diff);

# Taking into account the angle, calculate our change\_x

# and change\_y. Velocity is how fast the bullet travels.

velocity = 5

self.change\_x = math.cos(angle) \* velocity

self.change\_y = math.sin(angle) \* velocity

def update(self):

""" Move the bullet. """

# The floating point x and y hold our more accurate location.

self.floating\_point\_y += self.change\_y

self.floating\_point\_x += self.change\_x

# The rect.x and rect.y are converted to integers.

self.rect.y = int(self.floating\_point\_y)

self.rect.x = int(self.floating\_point\_x)

# If the bullet flies of the screen, get rid of it.

if self.rect.x < 0 or self.rect.x > SCREEN\_WIDTH or self.rect.y < 0 or self.rect.y > SCREEN\_HEIGHT:

self.kill()

# --- Create the window

# Initialize Pygame

pygame.init()

# Set the height and width of the screen

screen = pygame.display.set\_mode([SCREEN\_WIDTH, SCREEN\_HEIGHT])

# --- Sprite lists

# This is a list of every sprite. All blocks and the player block as well.

all\_sprites\_list = pygame.sprite.Group()

# List of each block in the game

block\_list = pygame.sprite.Group()

# List of each bullet

bullet\_list = pygame.sprite.Group()

# --- Create the sprites

for i in range(50):

# This represents a block

block = Block(BLUE)

# Set a random location for the block

block.rect.x = random.randrange(SCREEN\_WIDTH)

block.rect.y = random.randrange(SCREEN\_HEIGHT - 50)

# Add the block to the list of objects

block\_list.add(block)

all\_sprites\_list.add(block)

# Create a red player block

player = Player()

all\_sprites\_list.add(player)

# Loop until the user clicks the close button.

done = False

# Used to manage how fast the screen updates

clock = pygame.time.Clock()

score = 0

player.rect.x = SCREEN\_WIDTH / 2

player.rect.y = SCREEN\_HEIGHT / 2

# -------- Main Program Loop -----------

while not done:

# --- Event Processing

for event in pygame.event.get():

if event.type == pygame.QUIT:

done = True

elif event.type == pygame.MOUSEBUTTONDOWN:

# Fire a bullet if the user clicks the mouse button

# Get the mouse position

pos = pygame.mouse.get\_pos()

mouse\_x = pos[0]

mouse\_y = pos[1]

# Create the bullet based on where we are, and where we want to go.

bullet = Bullet(player.rect.x, player.rect.y, mouse\_x, mouse\_y)

# Add the bullet to the lists

all\_sprites\_list.add(bullet)

bullet\_list.add(bullet)

# --- Game logic

# Call the update() method on all the sprites

all\_sprites\_list.update()

# Calculate mechanics for each bullet

for bullet in bullet\_list:

# See if it hit a block

block\_hit\_list = pygame.sprite.spritecollide(bullet, block\_list, True)

# For each block hit, remove the bullet and add to the score

for block in block\_hit\_list:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

score += 1

print(score)

# Remove the bullet if it flies up off the screen

if bullet.rect.y < -10:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

# --- Draw a frame

# Clear the screen

screen.fill(WHITE)

# Draw all the spites

all\_sprites\_list.draw(screen)

# Go ahead and update the screen with what we've drawn.

pygame.display.flip()

# --- Limit to 20 frames per second

clock.tick(60)

pygame.quit()

1. Sesudah di modifikasi

import pygame

import random

import math

# Define some colors

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

BLUE = (0, 0, 255)

GREEN = (0, 255, 0)

PURPLE = (255, 0, 255)

SCREEN\_WIDTH = 500

SCREEN\_HEIGHT = 650

Gambar = pygame.image.load('Background.png')

Background = pygame.transform.smoothscale(Gambar,(SCREEN\_WIDTH,SCREEN\_HEIGHT))

# --- Classes

class Wall(pygame.sprite.Sprite):

"""This class represents the bar at the bottom that the player controls """

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

""" Constructor function """

# Call the parent's constructor

super().\_\_init\_\_()

# Make a BLUE wall, of the size specified in the parameters

self.image = pygame.Surface([width, height])

self.image.fill(BLUE)

# Make our top-left corner the passed-in location.

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

self.rect.y = y

self.rect.x = x

class Block(pygame.sprite.Sprite):

""" This class represents the block. """

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

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

self.meteor = pygame.image.load('Meteor.png')

self.image = pygame.transform.scale(self.meteor, (30, 30))

#self.image = pygame.Surface([20, 15])

#self.image.fill(color)

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

def reset\_pos(self):

""" Called when the block is 'collected' or falls off

the screen. """

self.rect.y = random.randrange(-700, -20)

self.rect.x = random.randrange(10, SCREEN\_WIDTH - 40)

def update(self):

""" Automatically called when we need to move the block. """

self.rect.y += 1

if self.rect.y > SCREEN\_HEIGHT + self.rect.height :

self.reset\_pos()

class bigBlock(pygame.sprite.Sprite):

""" This class represents the block. """

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

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

self.meteor = pygame.image.load('Meteor.png')

self.image = pygame.transform.scale(self.meteor, (65, 65))

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

def reset\_pos(self):

""" Called when the block is 'collected' or falls off

the screen. """

self.rect.y = random.randrange(-700, -20)

self.rect.x = random.randrange(10, SCREEN\_WIDTH - 80)

def update(self):

""" Automatically called when we need to move the block. """

self.rect.y += 2

if self.rect.y > SCREEN\_HEIGHT + self.rect.height :

self.reset\_pos()

class Player(pygame.sprite.Sprite):

""" This class represents the Player. """

def \_\_init\_\_(self, WIDTH, HEIGHT):

""" Set up the player on creation. """

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

self.width = WIDTH

self.height = HEIGHT

self.rocket = pygame.image.load('Pesawat.png')

self.image = pygame.transform.scale(self.rocket, (self.width, self.height))

#self.image = pygame.Surface([self.width, self.height])

#self.image.fill(RED)

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

self.rect.x = SCREEN\_WIDTH / 2 - self.width / 2

self.rect.y = SCREEN\_HEIGHT - self.height

self.change\_x = 0

self.change\_y = 0

self.walls = None

def changespeed(self,x,y):

self.change\_x += x

self.change\_y += y

def update(self):

# Move left/right

self.rect.x += self.change\_x

# Did this update cause us to hit a wall?

block\_hit\_list = pygame.sprite.spritecollide(self, self.walls, False)

for block in block\_hit\_list:

# If we are moving right, set our right side to the left side of

# the item we hit

if self.change\_x > 0:

self.rect.right = block.rect.left

else:

# Otherwise if we are moving left, do the opposite.

self.rect.left = block.rect.right

class Bullet(pygame.sprite.Sprite):

""" This class represents the bullet. """

def \_\_init\_\_(self, start\_x, start\_y, dest\_x, dest\_y):

# Call the parent class (Sprite) constructor

super().\_\_init\_\_()

# Set up the image for the bullet

self.image = pygame.Surface([4, 10])

self.image.fill(PURPLE)

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

# Move the bullet to our starting location

self.rect.x = start\_x

self.rect.y = start\_y

self.floating\_point\_x = start\_x

self.floating\_point\_y = start\_y

x\_diff = dest\_x - start\_x

y\_diff = dest\_y - start\_y

angle = math.atan2(y\_diff, x\_diff);

velocity = 5

self.change\_x = math.cos(angle) \* velocity

self.change\_y = math.sin(angle) \* velocity

def update(self):

""" Move the bullet. """

# The floating point x and y hold our more accurate location.

self.floating\_point\_y += self.change\_y

self.floating\_point\_x += self.change\_x

# The rect.x and rect.y are converted to integers.

self.rect.y = int(self.floating\_point\_y)

self.rect.x = int(self.floating\_point\_x)

# If the bullet flies of the screen, get rid of it.

if self.rect.x < 0 or self.rect.x > SCREEN\_WIDTH or self.rect.y < 0 or self.rect.y > SCREEN\_HEIGHT:

self.kill()

# --- membuat window

# inisialisasi Pygame

pygame.init()

#mengatur panjang lebar nya screen

screen = pygame.display.set\_mode([SCREEN\_WIDTH, SCREEN\_HEIGHT])

# --- Sprite lists

# This is a list of every sprite. All blocks and the player block as well.

all\_sprites\_list = pygame.sprite.Group()

# List of each block in the game

block\_list = pygame.sprite.Group()

# List of each bigBlock in the game

bigblock\_list = pygame.sprite.Group()

# List of each bullet

bullet\_list = pygame.sprite.Group()

# Make the walls. (x\_pos, y\_pos, width, height)

wall\_list = pygame.sprite.Group()

# --- membuat the sprites

for i in range(10):

# This represents a block

block = Block()

bigblock = bigBlock()

# Add the block to the list of objects

block\_list.add(block)

bigblock\_list.add(bigblock)

all\_sprites\_list.add(block,bigblock)

# mengatur lokasi block secara acak

block.reset\_pos()

bigblock.reset\_pos()

# --- make the tembok

wall = Wall(0, 0, 10, SCREEN\_HEIGHT)

wall\_list.add(wall)

all\_sprites\_list.add(wall)

wall = Wall(SCREEN\_WIDTH - 10, 0, 10, SCREEN\_HEIGHT)

wall\_list.add(wall)

all\_sprites\_list.add(wall)

# Create a red player block

player = Player(35, 80)

player.walls = wall\_list

all\_sprites\_list.add(player)

# Loop until the user clicks the close button.

done = False

# Used to manage how fast the screen updates

clock = pygame.time.Clock()

# This is a font we use to draw text on the screen (size 36)

font = pygame.font.Font(None, 36)

# Current score

score = 0

# Current level

level = 1

# -------- Main Program Loop -----------

while not done:

# --- Event Processing

for event in pygame.event.get():

if event.type == pygame.QUIT:

done = True

# Fire a bullet if the user clicks the mouse button

# Get the mouse position

pos = pygame.mouse.get\_pos()

mouse\_x = pos[0]

mouse\_y = pos[1]

bullet = Bullet(player.rect.x + player.width / 2, player.rect.y, mouse\_x, mouse\_y)

## pygame.mixer.music.load('Carefree.mp3')

## pygame.mixer.music.set\_endevent(pygame.constants.USEREVENT)

## pygame.mixer.music.play()

if event.type == pygame.MOUSEBUTTONDOWN:

all\_sprites\_list.add(bullet)

bullet\_list.add(bullet)

pygame.mixer.music.load('match1.wav')

pygame.mixer.music.play()

elif event.type == pygame.MOUSEBUTTONUP and level >= 6:

all\_sprites\_list.add(bullet)

bullet\_list.add(bullet)

if event.type == pygame.KEYDOWN:

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

player.changespeed(-3, 0)

elif event.key == pygame.K\_RIGHT:

player.changespeed(3, 0)

elif event.type == pygame.KEYUP:

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

player.changespeed(3, 0)

elif event.key == pygame.K\_RIGHT:

player.changespeed(-3, 0)

all\_sprites\_list.update()

## for bigblock in bigblock\_list:

## block\_hit\_list = pygame.sprite.spritecollide(bigblock, block\_list, False)

##

## for block in block\_hit\_list:

## block.reset\_pos()

for bullet in bullet\_list:

# See if it hit a block

block\_hit\_list = pygame.sprite.spritecollide(bullet, block\_list, True)

# For each block hit, remove the bullet and add to the score

for block in block\_hit\_list:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

score += 1

print(score)

#pygame.mixer.music.load('Ghost.mp3')

#pygame.mixer.music.play()

if bullet.rect.y < -5:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

for bullet in bullet\_list:

bigblock\_hit\_list = pygame.sprite.spritecollide(bullet, bigblock\_list, True)

for bigblock in bigblock\_hit\_list:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

score+= 2

print(score)

# Remove the bullet if it flies up off the screen

if bullet.rect.y < -5:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

for bullet in bullet\_list:

# See if it hit a block

wall\_hit\_list = pygame.sprite.spritecollide(bullet, wall\_list, False)

# For each block hit, remove the bullet and add to the score

for block in wall\_hit\_list:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

# Remove the bullet if it flies up off the screen

if bullet.rect.y < -10:

bullet\_list.remove(bullet)

all\_sprites\_list.remove(bullet)

# Check to see if all the blocks are gone.

# If they are, level up.

if len(block\_list) == 0 and len(bigblock\_list) == 0:

# Add one to the level

level += 1

# Add more blocks. How many depends on the level.

# Also, an 'if' statement could be used to change what

# happens customized to levels 2, 3, 4, etc.

for i in range(level \* 10):

# This represents a block

block = Block()

bigblock = bigBlock()

# Add the block to the list of objects

block\_list.add(block)

bigblock\_list.add(bigblock)

all\_sprites\_list.add(block,bigblock)

block.reset\_pos()

bigblock.reset\_pos()

# Clear the screen

#screen.fill(WHITE)

screen.blit(Background, (0,0))

# Draw all the spites

all\_sprites\_list.draw(screen)

text = font.render("Score: "+str(score), True, RED)

screen.blit(text, [10, 10])

text = font.render("Level: "+str(level), True, RED)

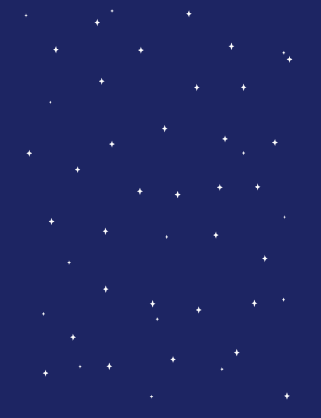
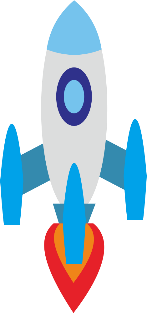
screen.blit(text, [10, 40])

pygame.display.flip()

clock.tick(70)

pygame.quit()

1. **Aset**
2. Gambar

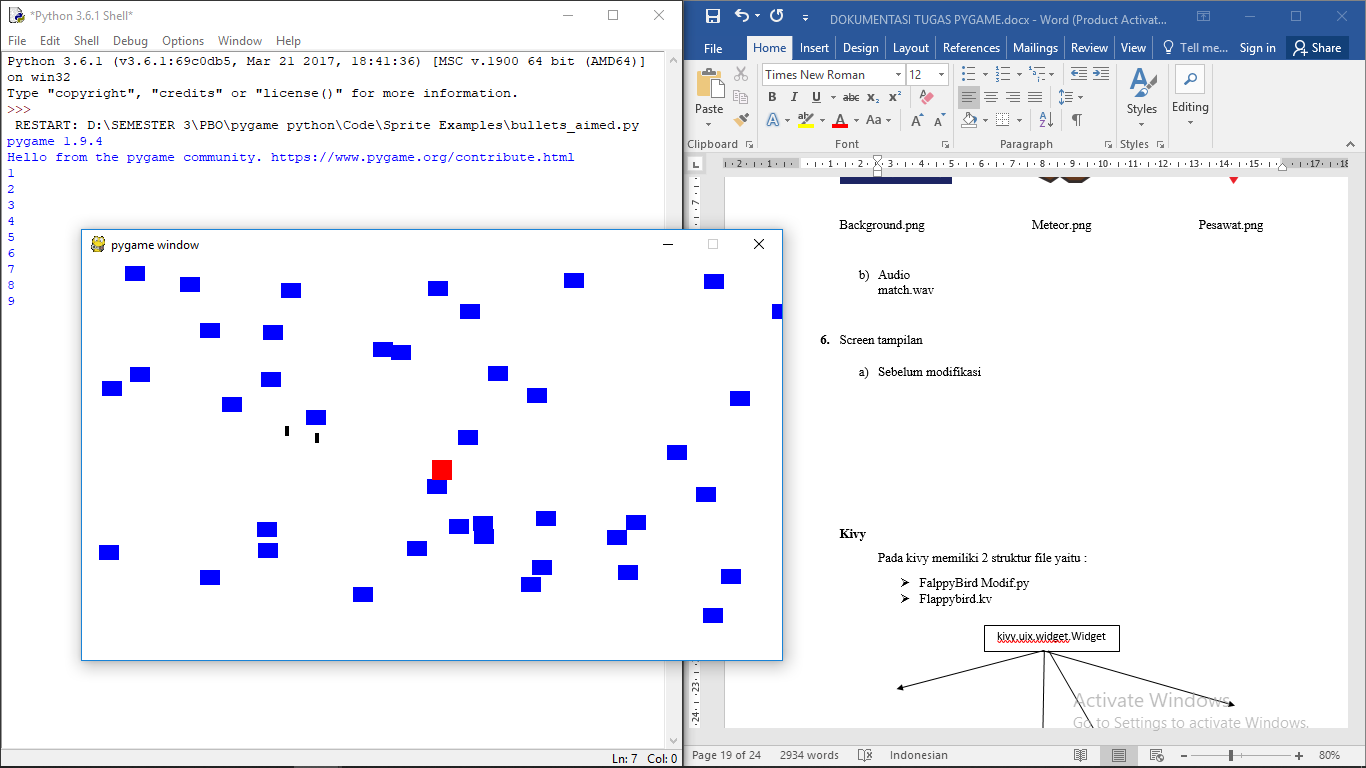
**  **

Background.png Meteor.png Pesawat.png

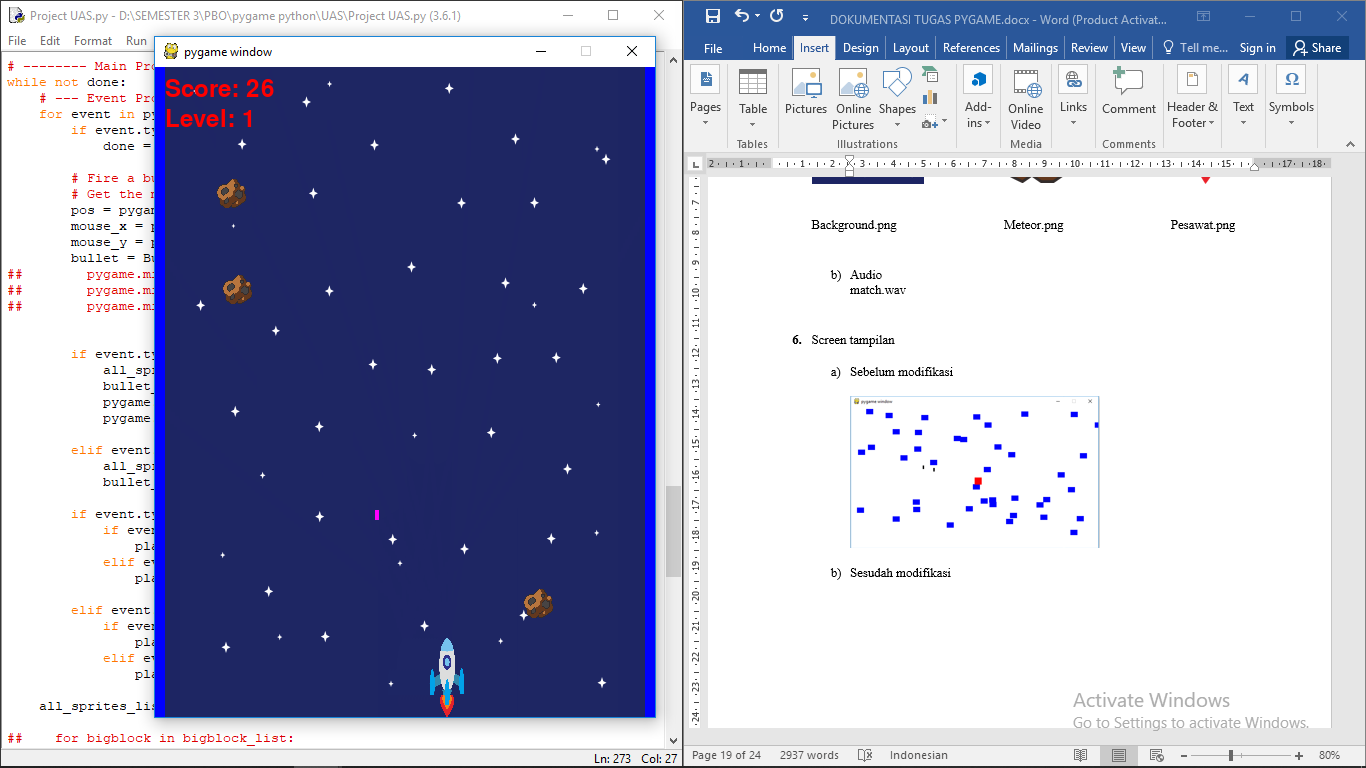
1. Audio

match.wav

1. Screen tampilan
2. Sebelum modifikasi



1. Sesudah modifikasi



**Kivy**

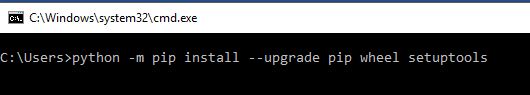
1. **Definisi Kivy**

Kivy adalah modul atau pustaka open source pada Python yang digunakan untuk membuat atau mengembangkan aplikasi seluler. Kivy dapat berjalan di Windows, OS X, Android, iOS, dan Raspberry Pi. Kita juga dapat menjalankan kode yang sama pada semua platform yang berbeda pula.

1. **Instalasi Kivy**

Buka cmd dengan cara tekan tombol “windows + r” ketik “cmd” lalu tekan enter.

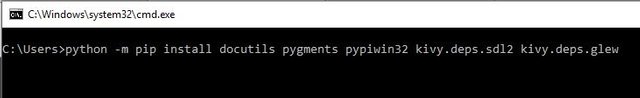
* 1. Pada tampilan cmd silahkan anda ketik perintah dibawah ini lalu tekan enter python -m pip install --upgrade pip wheel setuptools



Pastikan bahwa perangkat anda terhubung ke internet, lalu tunggu prosesnya hingga selesai, mungkin akan memakan waktu sekitar 5 menit tergantung pada kecepatan internet anda.

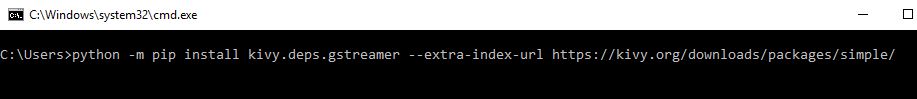
* 1. Jika langkah pertama sudah selesai ketik perintah berikut pada cmd pula, tunggu hingga proses selesai.

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



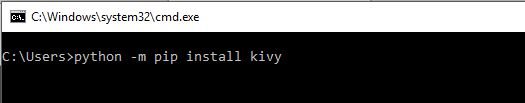
* 1. Sama seperti sebelumya

python -m pip install kivy.deps.gstreamer --extra-index-url https://kivy.org/downloads/packages/simple/

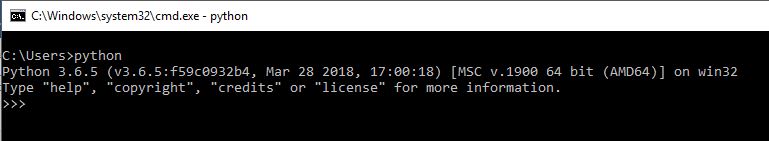


* 1. Langkah terakhir

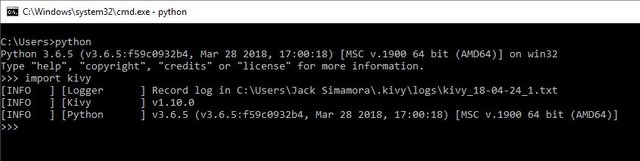
python -m pip install kivy



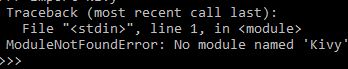
Untuk memastikan bahwa Kivy sudah terinstall dengan benar silahkan buka “cmd” lalu ketik python dan enter.



Ketik import kivy lalu enter



Jika kivy sudah terinstall dengan benar maka tampilannya akan sedikit seperti gambar yang diatas, jika belum maka akan seperti yang dibawah



Modul tidak ditemukan.

1. **Penjelasan OOP**

**Kivy**

Pada kivy memiliki 2 struktur file yaitu :

* FalppyBird Modif.py
* Flappybird.kv

kivy.uix.widget.Widget

|  |
| --- |
| Mcnay |
| * bird\_image * jump\_time * stop\_jumping * jumping * jump\_height * time\_jumped * switch\_to\_normal * on\_touch\_down * velocity\_x * velocity\_y * image * rect * \_keyboard * pos |
| + \_keyboard\_closed()  + switch\_to\_normal()  + stop\_jumping  + one\_touch\_down  + \_on\_keyboard\_down  + update |

|  |
| --- |
| Obstacle |
| * gap * pos |
| + update()  + update\_position() |

|  |
| --- |
| FlappyBirdGame |
| * mcnay * background * obstacles |
| + remove\_obastacle()  + new\_obstacle()  + size\_callback  + update |

|  |
| --- |
| Backgroud |
| * image\_one * image\_two * velocity |
| + update()  +update\_position() |

kivy.app.App

|  |
| --- |
| FlappyBirdApp |
|  |
| + build () |

Konsep OOP :

Class Mcnay, FlappyBirdGame, Obstacle dan Background merupakan inheritance dari class bawaan kivy yaitu kivy.uix.widget.Widget.

Sedangkan class FlappyBirdApp merupakan inheritance dari class bawaan kivy yaitu kivy.app.App.

Public Variable untuk menambahkan musik pada game

sfx\_flap = SoundLoader.load('flap.wav')

sfx\_score = SoundLoader.load('score.wav')

sfx\_die = SoundLoader.load('die.wav')

1. Class Background merupakan turunan dari class widget uix , memiliki method:
2. update

Untuk mengatur posisi background

1. update\_position

Untuk mengatur perubahan posisi pada backgroud

1. Class Mcnay merupakan turunan dari class widget uix, memiliki method :
2. Private method \_keyboard\_closed

Untuk mengatur saat keluar

1. switch\_to\_normal

Mengatur saat bird dalam keadaan normal dan tidak terbang

1. stop\_jumping

mengatur saat bird berhenti terbang

1. on\_touch\_down

mengatur saat keyboard disentuh dan bird terbang

1. \_on\_keyboard\_down

mengatur saat tidak menyentuh keyboard

1. update

mengatur posisi dan kecepatan saat berhenti terbang

1. Class Obstacle merupakan turunan dari class widget uix, memiliki method :
2. update\_position

mengatur lebar obstacle (jalan saat bird masuk)

1. update

mengatur posisi obstacle

1. Class FlappyBirdGame merupakan turunan dari class widget uix, memiliki method:
2. remove\_obstacle

menghapus obstacle

1. new\_obstacle

membuat obstacle baru dengan merubah lebar dan tinggi

1. size\_callback

mengupdate posisi obstacle

1. update

untuk mengupdate posisi obstacle saat bird telah melewatinya, mengatur score yang bertambah dan mengupdate background

1. Class App merupakan base class untuk membuat Aplikasi Kivy. Untuk menginisialisasi app dengan widget tree maka harus meng override App.build method di class app dan me-return widget tree yang telah dibangun.
2. Lalu jalankan program dengan menggunakan

if \_\_name\_\_ == "\_\_main\_\_":

FlappyBirdApp().run()

1. **Program**
2. Sebelum modifikasi

FlappyBird.py

import kivy

kivy.require("1.8.0")

from random import randint

import sys

from kivy.properties import NumericProperty, ReferenceListProperty, BooleanProperty, ObjectProperty, ListProperty

from kivy.uix.image import Image

from kivy.vector import Vector

from kivy.app import App

from kivy.clock import Clock

from kivy.config import Config

from kivy.core.window import Window

from kivy.uix.widget import Widget

class Background(Widget):

image\_one = ObjectProperty(Image())

image\_two = ObjectProperty(Image())

velocity\_x = NumericProperty(0)

velocity\_y = NumericProperty(0)

velocity = ReferenceListProperty(velocity\_x, velocity\_y)

def update(self):

self.image\_one.pos = Vector(\*self.velocity) + self.image\_one.pos

self.image\_two.pos = Vector(\*self.velocity) + self.image\_two.pos

if self.image\_one.right <= 0:

self.image\_one.pos = (self.width, 0)

if self.image\_two.right <= 0:

self.image\_two.pos = (self.width, 0)

def update\_position(self):

self.image\_one.pos = (0, 0)

self.image\_two.pos = (self.width, 0)

class Mcnay(Widget):

bird\_image = ObjectProperty(Image())

jump\_time = NumericProperty(0.3)

jump\_height = NumericProperty(95)

time\_jumped = NumericProperty(0)

jumping = BooleanProperty(False)

velocity\_x = NumericProperty(0)

velocity\_y = NumericProperty(0)

normal\_velocity\_x = NumericProperty(0)

normal\_velocity\_y = NumericProperty(0)

velocity = ReferenceListProperty(velocity\_x, velocity\_y)

normal\_velocity = ReferenceListProperty(normal\_velocity\_x, normal\_velocity\_y)

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

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

if Config.getdefault('input', 'keyboard', False):

self.\_keyboard = Window.request\_keyboard(

self.\_keyboard\_closed, self, 'text')

self.\_keyboard.bind(on\_key\_down=self.\_on\_keyboard\_down)

def \_keyboard\_closed(self):

self.\_keyboard.unbind(on\_key\_down=self.\_on\_keyboard\_down)

self.\_keyboard = None

def switch\_to\_normal(self, dt):

self.bird\_image.source = "images/flappyup.png"

Clock.schedule\_once(self.stop\_jumping, self.jump\_time \* (4.0 / 5.0))

def stop\_jumping(self, dt):

self.jumping = False

self.bird\_image.source = "images/flappy.png"

self.velocity\_y = self.normal\_velocity\_y

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

self.jumping = True

self.bird\_image.source = "images/flappynormal.png"

self.velocity\_y = self.jump\_height / (self.jump\_time \* 60.0)

Clock.unschedule(self.stop\_jumping)

Clock.schedule\_once(self.switch\_to\_normal, self.jump\_time / 5.0)

def \_on\_keyboard\_down(self, keyboard, keycode, text, modifiers):

self.on\_touch\_down(None)

def update(self):

self.pos = Vector(\*self.velocity) + self.pos

if self.pos[1] <= 104:

Clock.unschedule(self.stop\_jumping)

self.bird\_image.source = "images/flappynormal.png"

self.pos = (self.pos[0], 104)

class Obstacle(Widget):

gap\_top = NumericProperty(0)

gap\_size = NumericProperty(150)

velocity\_x = NumericProperty(0)

velocity\_y = NumericProperty(0)

velocity = ReferenceListProperty(velocity\_x, velocity\_y)

marked = BooleanProperty(False)

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

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

def update\_position(self):

self.gap\_top = randint(self.gap\_size + 112, self.height)

def update(self):

self.pos = Vector(\*self.velocity) + self.pos

class FlappyBirdGame(Widget):

mcnay = ObjectProperty(Mcnay())

background = ObjectProperty(Background())

obstacles = ListProperty([])

score = NumericProperty(0)

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

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

self.mcnay.normal\_velocity = [0, -4]

self.mcnay.velocity = self.mcnay.normal\_velocity

self.background.velocity = [-2, 0]

self.bind(size=self.size\_callback)

def remove\_obstacle(self):

self.remove\_widget(self.obstacles[0])

self.obstacles = self.obstacles[1:]

def new\_obstacle(self, remove=True):

if remove:

self.remove\_obstacle()

new\_obstacle = Obstacle()

new\_obstacle.height = self.height

new\_obstacle.x = self.width

new\_obstacle.update\_position()

new\_obstacle.velocity = [-3, 0]

self.add\_widget(new\_obstacle)

self.obstacles = self.obstacles + [new\_obstacle]

def size\_callback(self, instance, value):

for obstacle in self.obstacles:

obstacle.height = value[1]

obstacle.update\_position()

self.background.size = value

self.background.update\_position()

def update(self, dt):

self.mcnay.update()

self.background.update()

# Loop through and update obstacles. Replace obstacles which went off the screen.

for obstacle in self.obstacles:

obstacle.update()

if obstacle.x < self.mcnay.x and not obstacle.marked:

obstacle.marked = True

self.score += 1

self.new\_obstacle(remove=False)

if len(self.obstacles) == 0:

self.new\_obstacle(remove=False)

elif self.obstacles[0].x < 0:

self.remove\_obstacle()

# If obstacles is emply

# See if the player collides with any obstacles

for obstacle in self.obstacles:

if self.mcnay.collide\_widget(Widget(pos=(obstacle.x, obstacle.gap\_top + 20), size=(obstacle.width, obstacle.height - obstacle.gap\_top))):

# This will be replaced later on

sys.exit()

if self.mcnay.collide\_widget(Widget(pos=(obstacle.x, 0), size=(obstacle.width, obstacle.gap\_top - obstacle.gap\_size))):

# This will also be replaced

sys.exit()

class FlappyBirdApp(App):

def build(self):

game = FlappyBirdGame()

Clock.schedule\_interval(game.update, 1.0/60.0)

return game

if \_\_name\_\_ == "\_\_main\_\_":

FlappyBirdApp().run()

flappybird.kv

#:kivy 1.8.0

<FlappyBirdGame>:

mcnay: mcnay

background: background

canvas:

Color:

rgb: 137 / 255.0, 228 / 255.0, 135 / 255.0

Background:

id: background

pos: root.pos

Mcnay:

id: mcnay

center\_x: root.center\_x

center\_y: root.center\_y

Label:

font\_size: 70

center\_x: root.center\_x

top: root.top - 30

markup: True

text: "[color=ffffff]" + str(root.score) + "[/color]"

<Background>:

image\_one: image\_one

image\_two: image\_two

Image:

id: image\_one

source: "images/background.png"

pos: 0, 0

size: 800, 600

Image:

id: image\_two

source: "images/background.png"

pos: root.width, 0

size: 800, 600

<Obstacle>:

width: 30

canvas:

Color:

rgb: 21 / 255.0, 180 / 255.0, 39 / 255.0

Rectangle:

pos: self.x, self.gap\_top + 20

size: self.width, self.height - self.gap\_top

Rectangle:

pos: self.x, 112

size: self.width, self.gap\_top - self.gap\_size - 112

Image:

source: "images/pipe\_bottom.png"

center\_x: root.center\_x

y: root.gap\_top - 20

Image:

source: "images/pipe\_top.png"

center\_x: root.center\_x

y: root.gap\_top - root.gap\_size - 40

<Mcnay>:

bird\_image: image

size: 50, 50

Image:

id: image

source: "images/flappy.png"

size: root.size

pos: root.pos

1. Sesudah modifikasi

FlappyBird Modif.py

import kivy

kivy.require("1.8.0")

from random import randint

import sys

from kivy.properties import NumericProperty, ReferenceListProperty, BooleanProperty, ObjectProperty, ListProperty

from kivy.uix.image import Image

from kivy.vector import Vector

from kivy.app import App

from kivy.clock import Clock

from kivy.config import Config

from kivy.core.window import Window

from kivy.uix.widget import Widget

from kivy.core.audio import SoundLoader

sfx\_flap = SoundLoader.load('flap.wav')

sfx\_score = SoundLoader.load('score.wav')

sfx\_die = SoundLoader.load('die.wav')

class Background(Widget):

image\_one = ObjectProperty(Image())

image\_two = ObjectProperty(Image())

velocity\_x = NumericProperty(0)

velocity\_y = NumericProperty(0)

velocity = ReferenceListProperty(velocity\_x, velocity\_y)

def update(self):

self.image\_one.pos = Vector(\*self.velocity) + self.image\_one.pos

self.image\_two.pos = Vector(\*self.velocity) + self.image\_two.pos

if self.image\_one.right <= 0:

self.image\_one.pos = (self.width, 0)

if self.image\_two.right <= 0:

self.image\_two.pos = (self.width, 0)

def update\_position(self):

self.image\_one.pos = (0, 0)

self.image\_two.pos = (self.width, 0)

class Mcnay(Widget):

bird\_image = ObjectProperty(Image())

jump\_time = NumericProperty(0.3)

jump\_height = NumericProperty(65)

time\_jumped = NumericProperty(0)

jumping = BooleanProperty(False)

velocity\_x = NumericProperty(0)

velocity\_y = NumericProperty(0)

normal\_velocity\_x = NumericProperty(0)

normal\_velocity\_y = NumericProperty(0)

velocity = ReferenceListProperty(velocity\_x, velocity\_y)

normal\_velocity = ReferenceListProperty(normal\_velocity\_x, normal\_velocity\_y)

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

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

if Config.getdefault('input', 'keyboard', False):

self.\_keyboard = Window.request\_keyboard(self.\_keyboard\_closed, self, 'text')

self.\_keyboard.bind(on\_key\_down=self.\_on\_keyboard\_down)

def \_keyboard\_closed(self):

self.\_keyboard.unbind(on\_key\_down=self.\_on\_keyboard\_down)

self.\_keyboard = None

sfx\_die.play()

def switch\_to\_normal(self, dt):

self.bird\_image.source = "images/normal.png"

Clock.schedule\_once(self.stop\_jumping, self.jump\_time \* (3.0 / 5.0))

def stop\_jumping(self, dt):

self.jumping = False

self.bird\_image.source = "images/normal.png"

self.velocity\_y = self.normal\_velocity\_y

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

self.jumping = True

self.bird\_image.source = "images/normal.png"

self.velocity\_y = self.jump\_height / (self.jump\_time \* 60.0)

Clock.unschedule(self.stop\_jumping)

Clock.schedule\_once(self.switch\_to\_normal, self.jump\_time / 5.0)

sfx\_flap.play()

def \_on\_keyboard\_down(self, keyboard, keycode, text, modifiers):

self.on\_touch\_down(None)

def update(self):

self.pos = Vector(\*self.velocity) + self.pos

if self.pos[1] <= 104:

Clock.unschedule(self.stop\_jumping)

self.bird\_image.source = "images/normal.png"

self.pos = (self.pos[0], 104)

class Obstacle(Widget):

gap\_top = NumericProperty(0)

gap\_size = NumericProperty(240)

velocity\_x = NumericProperty(0)

velocity\_y = NumericProperty(0)

velocity = ReferenceListProperty(velocity\_x, velocity\_y)

marked = BooleanProperty(False)

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

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

def update\_position(self):

self.gap\_top = randint(self.gap\_size-40, self.height)

def update(self):

self.pos = Vector(\*self.velocity) + self.pos

class FlappyBirdGame(Widget):

mcnay = ObjectProperty(Mcnay())

background = ObjectProperty(Background())

obstacles = ListProperty([])

score = NumericProperty(0)

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

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

self.mcnay.normal\_velocity = [0, -4]

self.mcnay.velocity = self.mcnay.normal\_velocity

self.background.velocity = [-2, 0]

self.bind(size=self.size\_callback)

def remove\_obstacle(self):

self.remove\_widget(self.obstacles[0])

self.obstacles = self.obstacles[1:]

def new\_obstacle(self, remove=True):

if remove:

self.remove\_obstacle()

new\_obstacle = Obstacle()

new\_obstacle.height = self.height

new\_obstacle.x = self.width

new\_obstacle.update\_position()

new\_obstacle.velocity = [-2, 0]

self.add\_widget(new\_obstacle)

self.obstacles = self.obstacles + [new\_obstacle]

def size\_callback(self, instance, value):

for obstacle in self.obstacles:

obstacle.height = value[1]

obstacle.update\_position()

self.background.size = value

self.background.update\_position()

def update(self, dt):

self.mcnay.update()

self.background.update()

# Loop through and update obstacles. Replace obstacles which went off the screen.

for obstacle in self.obstacles:

obstacle.update()

if obstacle.x < self.mcnay.x and not obstacle.marked:

obstacle.marked = True

self.score += 1

self.new\_obstacle(remove=False)

sfx\_score.play()

if len(self.obstacles) == 0:

self.new\_obstacle(remove=False)

elif self.obstacles[0].x < 0:

self.remove\_obstacle()

# If obstacles is emply

# See if the player collides with any obstacles

for obstacle in self.obstacles:

if self.mcnay.collide\_widget(Widget(pos=(obstacle.x, obstacle.gap\_top + 20), size=(obstacle.width, obstacle.height - obstacle.gap\_top))):

# This will be replaced later on

sys.exit()

if self.mcnay.collide\_widget(Widget(pos=(obstacle.x, 0), size=(obstacle.width, obstacle.gap\_top - obstacle.gap\_size))):

# This will also be replaced

sys.exit()

class FlappyBirdApp(App):

def build(self):

game = FlappyBirdGame()

Clock.schedule\_interval(game.update, 1.0/60.0)

return game

if \_\_name\_\_ == "\_\_main\_\_":

FlappyBirdApp().run()

flappybird.kv

#:kivy 1.8.0

<FlappyBirdGame>:

mcnay: mcnay

background: background

canvas:

Color:

rgb: 137 / 255.0, 228 / 255.0, 135 / 255.0

Background:

id: background

pos: root.pos

Mcnay:

id: mcnay

center\_x: root.center\_x

center\_y: root.center\_y

Label:

font\_size: 30

center\_x: root.center\_x

right: root.right-650

markup: True

text: "[color=ffffff]" + str("Score :")+str(root.score) + "[/color]"

<Background>:

image\_one: image\_one

image\_two: image\_two

Image:

id: image\_one

source: "images/background.png"

pos: 0, 0

size: 1920,1080

Image:

id: image\_two

source: "images/background.png"

pos: root.width, 0

size: 1920,1080

<Obstacle>:

width: 65

canvas:

Color:

rgb: 21 / 255.0, 180 / 255.0, 39 / 255.0

Rectangle:

pos: self.x, self.gap\_top + 20

size: self.width, self.height - self.gap\_top

Rectangle:

pos: self.x,112

size: self.width, self.gap\_top - self.gap\_size - 112

Image:

source: "images/pipe\_bottom.png"

center\_x: root.center\_x

y: root.gap\_top - 20

Image:

source: "images/pipe\_top.png"

center\_x: root.center\_x

y: root.gap\_top - root.gap\_size - 40

<Mcnay>:

bird\_image: image

size: 50, 50

Image:

id: image

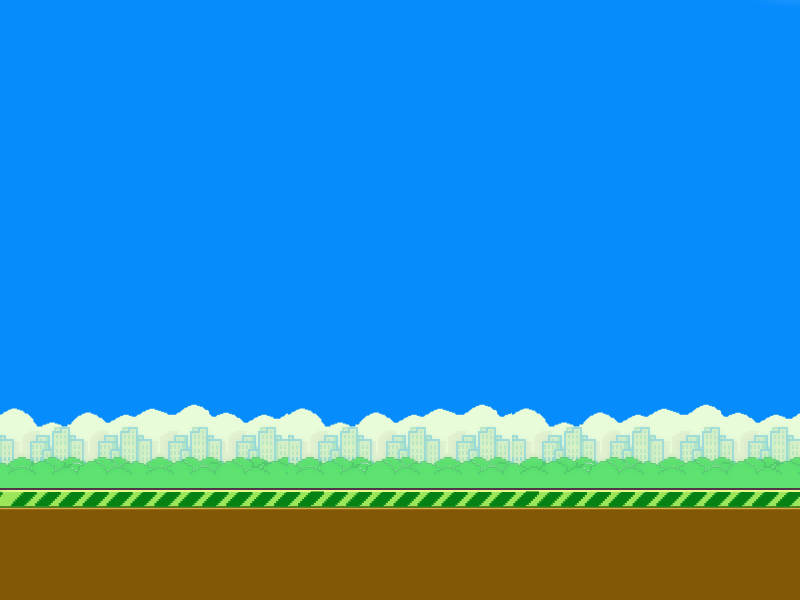
source: "images/normal.png"

size: root.size

pos: root.pos

Sumber code : <https://github.com/undercase/FlappyKivy>

1. **Aset**
2. Sebelum modifikasi



background.png



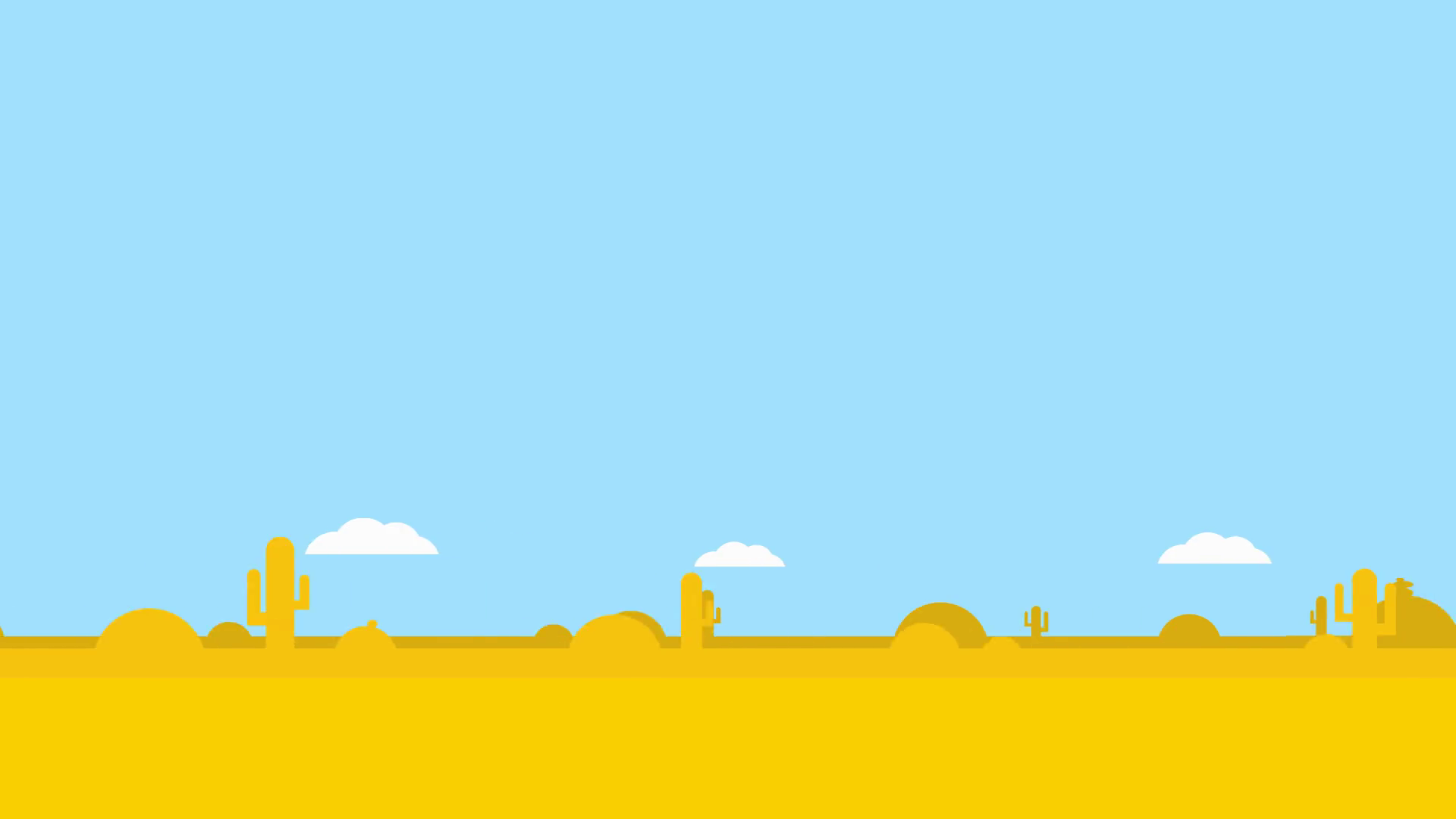
flappynormal.png flappyup.png flappy.png



pipe\_top.png pipe\_bottom.png

1. Sesudah modifikasi

Gambar



background.png

C:\Users\Laptop\AppData\Local\Microsoft\Windows\INetCache\Content.Word\pipe_top.png C:\Users\Laptop\AppData\Local\Microsoft\Windows\INetCache\Content.Word\pipe_bottom.png



normal.png pipe\_top.png pipe\_bottom.png

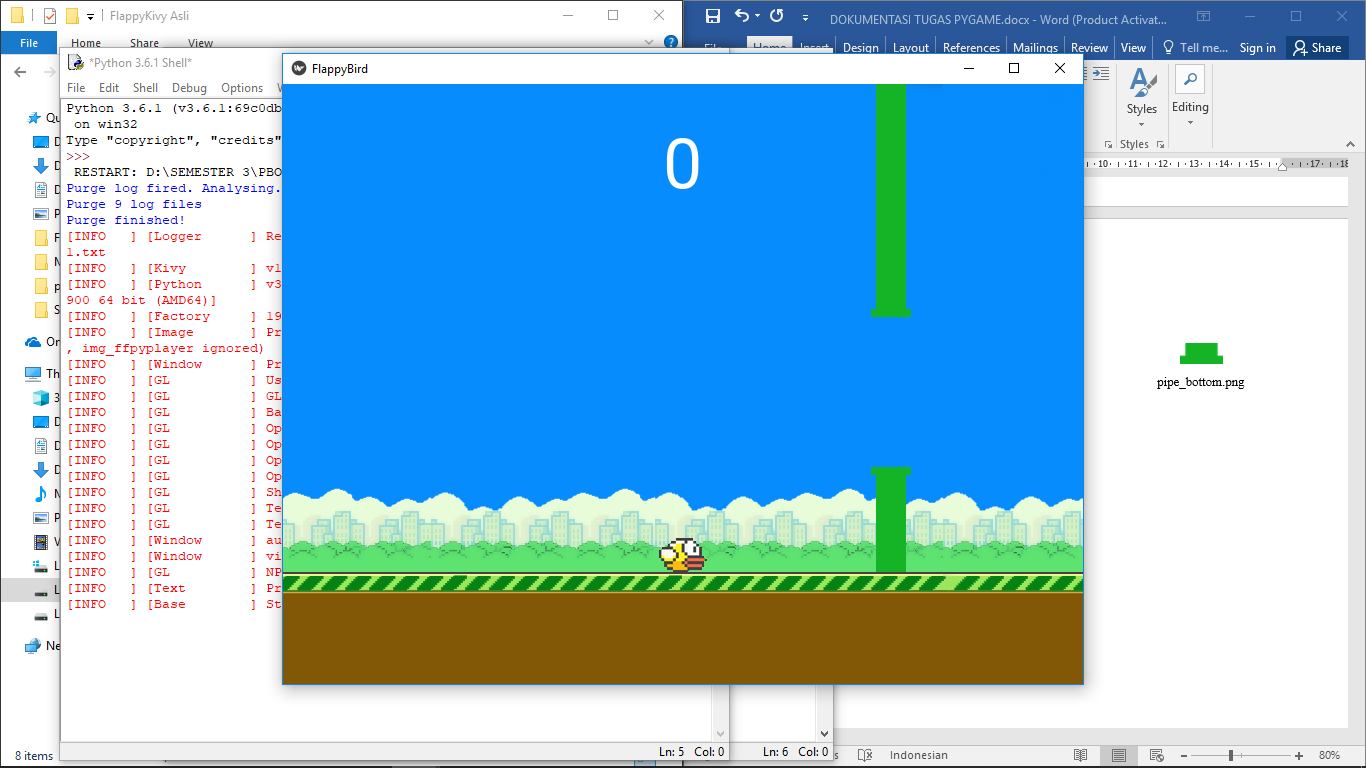
Audio

die.wav

flap.wav

score.wav

1. Screen tampilan
2. Sebelum modifikasi



1. Sesudah modifikasi

