

LAPORAN
PEMROGRAMAN BERORIENTASI OBJEK
“Pygame dan Kivy”



disusun oleh:

Gita Dharma Ramadhani	170411100002
Rosyaifa Meifani Pratami	170411100080

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA
2018/2019

PYGAME

A. Penjelasan

Pygame merupakan salah satu *library* dari bahasa pemrograman Python yang gratis dan *Open Source* untuk membuat aplikasi multimedia seperti permainan yang dibangun di atas perpustakaan SDL yang sangat baik. Seperti SDL, Pygame sangat portabel, karena mendukung Windows, Linux, Mac OS X, BeOS, FreeBSD, NetBSD, OpenBSD, BSD/OS, Olaris, IRIX, dan QNX. Selain itu, subset Pygame untuk Android tersedia. Pygame tidak membutuhkan OpenGL dan bisa menggunakan DirectX, WinDIB, X11, Linux framebuffer, dan berbagai API lain untuk merender grafis. Hal ini memastikan banyak user dapat memainkan *game* yang mereka inginkan.

B. Cara Instalasi Pygame

a) Melalui pygame.whl

Agar modul pygame dapat digunakan untuk pembuatan game, maka terlebih dahulu dilakukan instalasi Pygame. Tahapan-tahapan yang harus dilakukan untuk instalasi pygame ini adalah :

1. 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.

2. Rubah file pygame.whl menjadi .zip

Rubah extension file library pygame lalu extract isi library tersebut.

3. Copy file pygame yang dibutuhkan

- Masuk ke dalam directory python
(C:\Users\#username\AppData\Local\Programs\Python\Python35-32)
- Masuk kedalam folder “include” dan buat folder baru bernama “pygame”.
- Di dalam folder hasil extract file library pygame yang sudah didownload, masuk ke “pygame-1.9.4\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`

- Kembali ke folder hasil extract file library pygame tadi, copy folder “pygame” dan “pygame-1.9.4.dist-info” kedalam:

`C:\Users\#username\AppData\Local\Programs\Python\Python35-32\Lib\site-packages`

4. Cek hasil instalasi

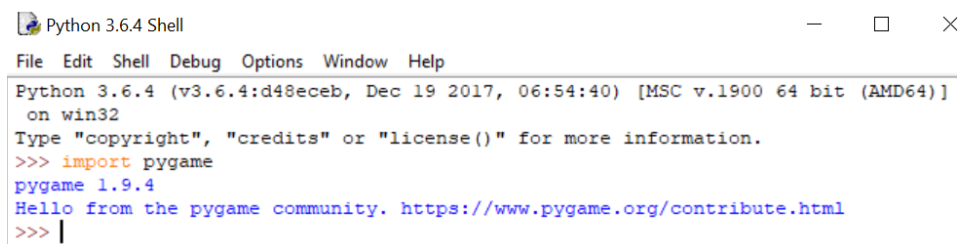
Buka IDLE Python lalu lakukan perintah “import pygame”, jika tidak ada tulisan error maka pygame sudah berhasil terinstall.

b) Melalui pip

1. Instal Python yang dapat diunduh dari python.org. Disarankan python 3.6.1 atau yang terbaru, karena lebih mudah digunakan untuk pemula.
2. Pastikan ketika menginstal python3.6 dengan opsi "Tambahkan python 3.6 to PATH" yang dipilih. Ini berarti bahwa python, dan pip akan bekerja dari baris perintah.
3. Bukalah command prompt kemudian ketik seperti dibawah ini

```
py -m pip install -U pygame --user
```

4. Untuk melihat apakah berfungsi, silahkan masuk ke *shell* python (IDLE Python) dengan mengetik perintah python3:

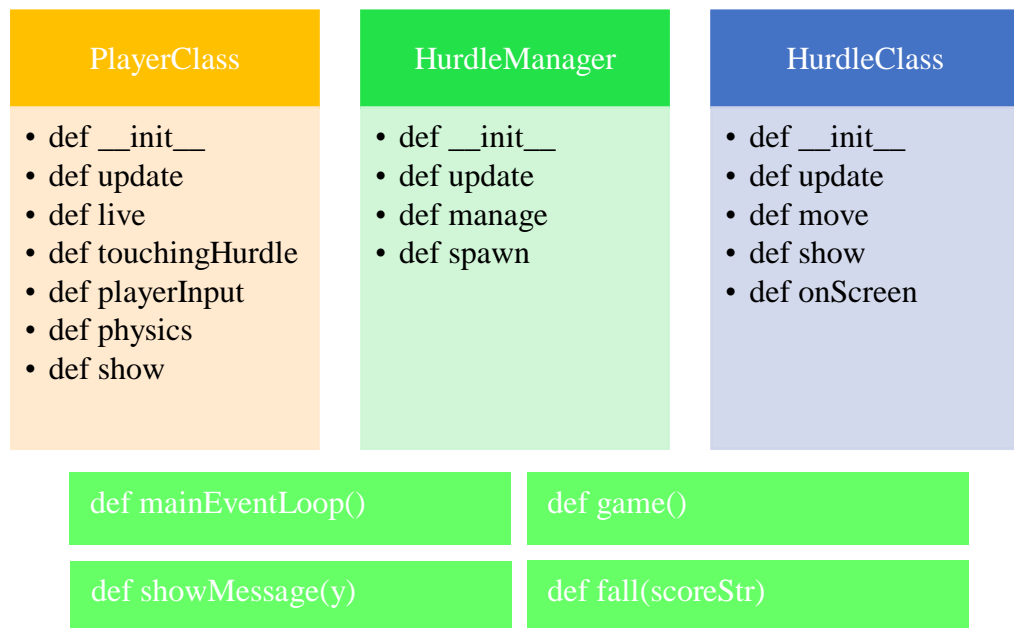


```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> import pygame
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> |
```

5. Jika tidak terjadi error, maka modul pygame sudah terinstal dengan benar. Namun jika terjadi error, artinya modul pygame belum terinstall.

C. Penjelasan OOP Pada Program Pygame

a) Diagram Kelas dan Objek Pada Program Pygame



Global Variable :

- + windowX, sebagai variabel untuk menginisiasi lebar dari layar window.
- + WindowY, sebagai variabel untuk menginisiasi tinggi dari layar window.
- + Nyawa, sebagai variabel untuk menginisiasi berapa nyawa yang akan dipakai untuk pemain jika menyentuh gawang atau hurdle.
- + clock, sebagai variabel untuk menginisiasi waktu pada method pygame yang tersedia.
- + Window, sebagai variabel untuk menginisiasi lebar dan tinggi pada layar window sehingga menggunakan variabel windowX dan windowY untuk mengatur lebar dan tinggi layar.
- + Ground, sebagai variabel untuk mengatur tinggi ground yang didefinisikan windowX dikurangi 70.
- + gravity , sebagai variabel untuk menginisiasi nilai 1 yang digunakan untuk mengatur kecepatan pemain.
- + groundImg, sebagai variabel yang digunakan untuk mengatur skala dari gambar ground.
- + font1, sebagai variabel untuk mengatur font pertama.

- + font2, sebagai variabel untuk mengatur font kedua.
- + deathMessage1, sebagai variabel untuk mengatur teks pertama yang ditampilkan ketika game selesai.
- + deathMessage1Shadow, sebagai variabel untuk mengatur bayangan teks pertama yang ditampilkan ketika game selesai.
- + deathMessage2, sebagai variabel untuk mengatur teks kedua yang ditampilkan ketika game selesai.
- + deathMessage2Shadow, sebagai variabel untuk mengatur bayangan teks kedua yang ditampilkan ketika game selesai.
- + message1Rect, sebagai variabel untuk menaruh teks pertama pada kotak atau rectangle sebagai method yang telah disediakan oleh pygame.
- + message1x, sebagai variabel untuk mengatur tinggi dan lebar dari message1Rect.
- + message2Rect, sebagai variabel untuk menaruh teks kedua pada kotak atau rectangle sebagai method yang telah disediakan oleh pygame.
- + message2x, sebagai variabel untuk mengatur tinggi dan lebar dari message2Rect.

b) Penjelasan dan Konsep OOP

- Penjelasan Diagram

- 1) **def __init__** pada kelas **PlayerClass** berfungsi untuk mengorganisasikan serta mendefinisikan kelas pemain.
- 2) **def update** pada kelas **PlayerClass** berfungsi untuk memperbarui status pemain jika menyentuh gawang atau hurdle.
- 3) **def live** pada kelas **PlayerClass** berfungsi untuk menginisiasi status hidup pemain.
- 4) **def touchingHurdle** pada kelas **PlayerClass** berfungsi untuk mengatur jarak pemain saat menyentuh gawang atau hurdle.
- 5) **def playerInput** pada kelas **PlayerClass** berfungsi untuk mengatur tombol yang ditekan agar pemain dapat melompati gawang atau hurdle.

- 6) **def physics** pada kelas **PlayerClass** berfungsi untuk mengatur tinggi rendahnya lompatan pemain.
- 7) **def show** pada kelas **PlayerClass** berfungsi untuk mengatur asset atau gambar yang ditampilkan ketika pemain berjalan dan melompat serta mengatur kecepatan gawang atau hurdle yang harus dilewati pemain saat skor yang didapat oleh pemain semakin tinggi.
- 8) **def __init__** pada kelas **HurdleManager** berfungsi untuk mengorganisasikan serta me-manage gawang atau hurdle. Pada kelas ini lebih ditekankan pada kecepatan pergerakan gawang atau hurdle.
- 9) **def update** pada kelas **HurdleManager** berfungsi untuk memperbarui kecepatan gawang atau hurdle.
- 10) **def manage** pada kelas **HurdleManager** berfungsi untuk me-manage pengelompokan gawang atau hurdle pada list.
- 11) **def spawn** pada kelas **HurdleManager** berfungsi untuk mengatur lebar jarak dari gawang ke gawang ketika kecepatan gawang atau hurdle semakin naik. Pada method ini, terdapat pemanggilan kelas **HurdleClass**.
- 12) **def __init__** pada **HurdleClass** berfungsi untuk mengorganisasikan gawang atau hurdle. Pada kelas ini lebih ditekankan pada grafik serta perbaruan pada kecepatan gawang atau hurdle.
- 13) **def update** pada kelas **HurdleClass** berfungsi untuk memperbarui kecepatan gawang atau hurdle.
- 14) **def move** pada kelas **HurdleClass** berfungsi untuk mengatur perpindahan gawang atau hurdle.
- 15) **def show** pada kelas **HurdleClass** berfungsi untuk menampilkan gambar atau asset dari gawang serta mengatur lebar dan tinggi dari gawang tersebut.
- 16) **def onScreen** pada kelas **HurdleClass** berfungsi untuk menginisiasi tipe data Boolean True ketika jumlah dari lebar dan x lebih besar dari 0 dan menginisiasi tipe data Boolean False untuk lainnya.

- 17) **def mainEventLoop** berfungsi untuk mengatur event pada program agar ketika menekan tombol x dapat menutup program walau program sedang berjalan.
- 18) **def showMessage** berfungsi untuk mengatur teks yang tampil pada saat game selesai.
- 19) **def game** berfungsi sebagai inti dari program sehingga dalam method ini terdapat pengulangan while yang mengikutsertakan pemanggilan method penting lainnya.
- 20) **def fall** berfungsi untuk mengatur tombol ketika pemain jatuh yang berarti game selesai, sehingga dalam method ini terdapat pengulangan while dan beberapa pemanggilan method lainnya.

- **Kelas dan Objek**

Struktur data yang bisa kita gunakan untuk mendefinisikan objek yang menyimpan data bersama-sama nilai-nilai dan perilaku(behavior). Kelas adalah suatu entitas yang merupakan bentuk program dari suatu abstraksi untuk permasalahan dunia nyata, dan instans dari class yang merupakan realisasi dari beberapa objek. Contoh Kelas dan Objek pada program pygame :

- 1) Kelas PlayerClasss
- 2) Kelas HurdleManager
- 3) Kelas HurdleClass
- 4) player sebagai objek dari kelas PlayerClass
- 5) hurdleManager sebagai objek dari kelas HurdleManager
- 6) newHurdle yang terdapat pada kelas HurdleManager.spawn sebagai objek dari HurdleClass.

- **Override Enheritance**

Method atau properti yang sudah ada dimodifikasi atau diganti nilai properti atau methodnya agar sesuai yang diinginkan. Properti yang digunakan harus sama dengan properti di kelas anak yang

dimodifikasi, isi method boleh diganti. Contoh Override

Enheritance pada program pygame : -

- Multiple Enheritance

Pada Multiple Enheritance subclass dapat mewarisi atribut dan method lebih dari satu parent class. Contoh Multiple Enheritance pada program pygame : -

- Super Enheritance

Pada Multiple Enheritance subclass dapat menambah properti dari parent class. Contoh Super Enheritance pada program pygame : -

- Polymorphism with Function

Pada Polymorphism with Function, properti di masing-masing class memiliki nama yang sama namun isinya berbeda, dan pemanggilan kelas-kelas tersebut dengan menggunakan dengan menggunakan objek yang terdapat pada global fungsi. Contoh Polymorphism with Function pada program pygame : -

- Polymorphism with Class

Pada Polymorphism with Class, metode yang digunakan dengan cara tanpa mengetahui atau tidak peduli apa tipe kelas dari masing-masing objek ini. Pertama-tama dapat membuat perulangan for yang beriterasi melalui tuple objek. Contoh Polymorphism with Class pada program pygame : -

- Overloading

Overloading merupakan sebuah method dengan multiple cara, dengan beda parameter, dan berbeda dari fungsi yang didefinisikan. Operator overloading merupakan operator yang banyak bentuk. Contoh Overloading pada program pygame : -

- Encapsulation

Enkapsulasi (pembungkusan atau pengkapsulan) merupakan suatu cara untuk menyembunyikan suatu proses atau data didalam sistem aplikasi. Dengan enkapsulasi, pengguna dapat memilih *property* dan *method* apa saja yang boleh diakses, dan method yang tidak boleh diakses yaitu dengan menggunakan private method/properti dan public method/properti. Bersifat privat maka tidak dapat diakses di luar kelas dan bersifat public maka dapat diakses diluar kelas. Contoh enkapsulasi pada program pygame : -

- Magic Method

Magic method merupakan operator overloading yang dicapai dengan mendefinisikan metode khusus dalam definisi kelas. Contoh Magic Method pada program pygame : -

D. Program Pygame

a) Program Awal

```
File Edit Format Run Options Window Help
import pygame, sys
import random

windowX = 720
windowY = 360

pygame.init()
clock = pygame.time.Clock()
window = pygame.display.set_mode((windowX, windowY))

pygame.display.set_caption('Jumpy Thing!')

ground = windowY - 70

gravity = 1

class PlayerClass:
    def __init__(self, scale, imageChangeSpeed, terminalVelocity):
        self.run1 = pygame.transform.scale(pygame.image.load('duck1.png'), (14 * scale, 14 * scale))
        self.run2 = pygame.transform.scale(pygame.image.load('duck2.png'), (14 * scale, 14 * scale))
        self.run3 = pygame.transform.scale(pygame.image.load('duck1.png'), (14 * scale, 14 * scale))
        self.run4 = pygame.transform.scale(pygame.image.load('duck2.png'), (14 * scale, 14 * scale))
        self.run5 = pygame.transform.scale(pygame.image.load('duck1.png'), (14 * scale, 14 * scale))

        self.scale = scale
        self.imageChangeSpeed = imageChangeSpeed
        self.terminalVelocity = terminalVelocity

        self.height = 14 * scale
        self.width = 7 * scale

    dead = False

    def update(self):
        self.physics()

        if self.touchingHurdle():
            self.dead = True

        if not self.dead:
            self.playerInput()

        self.y += self.velocityY

        self.show()

    def touchingHurdle(self):
        for hurdle in hurdleManager.hurdleList:
            if self.x + self.width > hurdle.x:
                if self.x < hurdle.x + hurdle.width:
                    if self.y + self.height > hurdle.y:
                        return True

x = 100
y = 100

velocityY = 0

def playerInput(self):
    pressedKeys = pygame.key.get_pressed()

    if pressedKeys[pygame.K_SPACE]:
        if self.y + self.height == ground:
            self.velocityY -= 10
        else:
            self.velocityY -= gravity / 2
```

```

def physics(self):
    if self.dead:
        if self.y < windowHeight:
            self.velocityY += 1

    elif self.y + self.height < ground:
        if self.velocityY < self.terminalVelocity:
            self.velocityY += gravity

    elif self.velocityY > 0:
        self.velocityY = 0
        self.y = ground - self.height

runTick = 0

def show(self):
    if self.runTick <= self.imageChangeSpeed:
        img = self.run1
    elif self.runTick <= self.imageChangeSpeed * 2:
        img = self.run2
    elif self.runTick <= self.imageChangeSpeed * 3:
        img = self.run3
    elif self.runTick <= self.imageChangeSpeed * 4:
        img = self.run4
    else:
        img = self.run5

    self.runTick += 1

    if self.runTick >= self.imageChangeSpeed * 5:
        self.runTick = 0

    window.blit(img, (self.x, self.y))

player = PlayerClass(5, 6, 10)

class HurdleManager:
    def __init__(self, scale, spawnRange):
        self.img = pygame.transform.scale(pygame.image.load('hurdle.png'), (7 * scale, 7 * scale))

        self.spawnRange = spawnRange
        self.hurdleList = []
        self.scale = scale

    def update(self, doSpawn, moveSpeed):
        if doSpawn:
            self.spawn()
            self.manage(moveSpeed)

    def manage(self, moveSpeed):
        hurdles2 = []

        for hurdle in self.hurdleList:
            hurdle.update(moveSpeed)

            if hurdle.onScreen():
                hurdles2.append(hurdle)

        self.hurdleList = hurdles2

    spawnTick = 0

```

```

def spawn(self):
    if self.spawnTick >= self.spawnRange[1]:
        newHurdle = HurdleClass(windowX, self.img, 7 * self.scale, 15 * self.scale)
        self.hurdleList.append(newHurdle)
        self.spawnTick = 0

    elif self.spawnTick > self.spawnRange[0]:
        if random.randint(0, self.spawnRange[1] - self.spawnRange[0]) == 0:
            newHurdle = HurdleClass(windowX, self.img, 7 * self.scale, 15 * self.scale)
            self.hurdleList.append(newHurdle)
            self.spawnTick = 0

    self.spawnTick += 1

hurdleManager = HurdleManager(3, (30, 90))

class HurdleClass:
    def __init__(self, x, img, width, height):
        self.x = x
        self.img = img
        self.width = width
        self.height = height
        self.y = ground - height

    def update(self, moveSpeed):
        self.move(moveSpeed)
        self.show()

    def move(self, moveSpeed):
        self.x -= moveSpeed

    def show(self):
        window.blit(self.img, (self.x, self.y))

    def onScreen(self):
        if self.x + self.width > 0:
            return True
        else:
            return False

def mainEventLoop():
    for events in pygame.event.get():
        if events.type == pygame.KEYDOWN:
            if events.key == pygame.K_ESCAPE:
                quit()

groundImg = pygame.transform.scale(pygame.image.load('ground.png'), (windowX, int(windowY - ground)))

font1 = pygame.font.Font(('calibri.ttf'), 50)
font2 = pygame.font.Font(('calibri.ttf'), 40)
deathMessage1 = font1.render('You Fell Over!', True, (255, 255, 255))
deathMessage1Shadow = font1.render('You Fell Over!', True, (0, 0, 0))
deathMessage2 = font2.render('Press Space', True, (255, 255, 255))
deathMessage2Shadow = font2.render('Press Space', True, (0, 0, 0))

message1Rect = deathMessage1.get_rect()
message1x = windowX / 2 - message1Rect.width / 2

message2Rect = deathMessage2.get_rect()
message2x = windowX / 2 - message2Rect.width / 2

```

```

def showMessage(y):
    window.blit(deathMessage1, (message1x, y))
    window.blit(deathMessage1Shadow, (message1x + 5, y + 5))

    window.blit(deathMessage2, (message2x, y + message1Rect.height))
    window.blit(deathMessage2Shadow, (message2x, message1Rect.height + 5 + y))

score = {'gameScore': 0}

def game():
    player.update()
    while True:
        if player.dead:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
            fall(scoreStr)
        else:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
    mainEventLoop()
    window.fill((200, 240, 250))

    player.update()

    hurdleManager.update(True, score['gameScore'] / 50 + 3)
    window.blit(groundImg, (0, ground))

    clock.tick(60)

    scoreStr = font2.render(str(round(score['gameScore'])), True, (0, 0, 0))
    window.blit(scoreStr, (50, 50))

    pygame.display.update()

    score['gameScore'] += 0.1

def fall(scoreStr):
    space = 0
    while True:
        pressedKeys = pygame.key.get_pressed()

        oldSpace = space
        space = pressedKeys[pygame.K_SPACE]

        mainEventLoop()
        window.fill((200, 240, 250))

        player.update()

        hurdleManager.update(False, score['gameScore'] / 50 + 3)
        window.blit(groundImg, (0, ground))

        clock.tick(60)

        showMessage(50)

        window.blit(scoreStr, (50, 50))
        pygame.display.update()

        spaceEvent = space - oldSpace

```

```

if spaceEvent == 1:
    #Reset Everything

    hurdleManager.hurdleList = []
    player.velocityY = 0
    player.dead = False
    player.y = ground - player.height
    score['gameScore'] = 0

    break
else:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

```

```
game()
```

b) Program Setelah Modifikasi

```

File Edit Format Run Options Window Help
import pygame, sys
import random
|
windowX = 720
windowY = 360
nyawa = 3

pygame.init()
clock = pygame.time.Clock()
window = pygame.display.set_mode((windowX, windowY))

pygame.display.set_caption('Jumpy Thing!')

ground = windowY - 70

gravity = 1

class PlayerClass:
    def __init__(self, scale, imageChangeSpeed, terminalVelocity, nyawa):
        self.run1 = pygame.transform.scale(pygame.image.load('duck1.png'), (14 * scale, 14 * scale))
        self.run2 = pygame.transform.scale(pygame.image.load('duck2.png'), (14 * scale, 14 * scale))
        self.run3 = pygame.transform.scale(pygame.image.load('duck1.png'), (14 * scale, 14 * scale))
        self.run4 = pygame.transform.scale(pygame.image.load('duck2.png'), (14 * scale, 14 * scale))
        self.run5 = pygame.transform.scale(pygame.image.load('duck1.png'), (14 * scale, 14 * scale))

        self.nyawa = nyawa
        self.scale = scale
        self.imageChangeSpeed = imageChangeSpeed
        self.terminalVelocity = terminalVelocity

        self.height = 14 * scale
        self.width = 7 * scale
        self.tabrak = 17

    dead = False

```

```

def update(self):
    self.physics()

    if self.nyawa == 1:
        self.dead = True

    if self.touchingHurdle():
        self.tabrak = self.tabrak -1

    if self.tabrak==0:
        self.live()
        self.tabrak = 17

    if not self.dead:
        self.playerInput()

    self.y += self.velocityY

    self.show()

def live(self):
    self.nyawa-=1

def touchingHurdle(self):
    for hurdle in hurdleManager.hurdleList:
        if self.x + self.width > hurdle.x:
            if self.x < hurdle.x + hurdle.width:
                if self.y + self.height > hurdle.y:
                    return True

x = 100
y = 100

velocityY = 0

def playerInput(self):
    pressedKeys = pygame.key.get_pressed()

    if pressedKeys[pygame.K_SPACE]:
        if self.y + self.height == ground:
            self.velocityY -= 10
        else:
            self.velocityY -= gravity / 2

def physics(self):
    if self.dead:
        if self.y < windowHeight:
            self.velocityY += 1

    elif self.y + self.height < ground:
        if self.velocityY < self.terminalVelocity:
            self.velocityY += gravity

    elif self.velocityY > 0:
        self.velocityY = 0
        self.y = ground - self.height

runTick = 0

```

```

def show(self):
    if self.runTick <= self.imageChangeSpeed:
        img = self.run1
    elif self.runTick <= self.imageChangeSpeed * 2:
        img = self.run2
    elif self.runTick <= self.imageChangeSpeed * 3:
        img = self.run3
    elif self.runTick <= self.imageChangeSpeed * 4:
        img = self.run4
    else:
        img = self.run5

    self.runTick += 1

    if self.runTick >= self.imageChangeSpeed * 5:
        self.runTick = 0

    window.blit(img, (self.x, self.y))

player = PlayerClass(5, 6, 10, nyawa)

class HurdleManager:
    def __init__(self, scale, spawnRange):
        self.img = pygame.transform.scale(pygame.image.load('hurdle.png'), (7 * scale, 7 * scale))

        self.spawnRange = spawnRange
        self.hurdleList = []
        self.scale = scale

    def update(self, doSpawn, moveSpeed):
        if doSpawn:
            self.spawn()
        self.manage(moveSpeed)

    def manage(self, moveSpeed):
        hurdles2 = []

        for hurdle in self.hurdleList:
            hurdle.update(moveSpeed)

            if hurdle.onScreen():
                hurdles2.append(hurdle)

        self.hurdleList = hurdles2

    spawnTick = 0

    def spawn(self):
        if self.spawnTick >= self.spawnRange[1]:
            newHurdle = HurdleClass(windowX, self.img, 7 * self.scale, 15 * self.scale)
            self.hurdleList.append(newHurdle)
            self.spawnTick = 0

        elif self.spawnTick > self.spawnRange[0]:
            if random.randint(0, self.spawnRange[1] - self.spawnRange[0]) == 0:
                newHurdle = HurdleClass(windowX, self.img, 7 * self.scale, 15 * self.scale)
                self.hurdleList.append(newHurdle)
                self.spawnTick = 0

        self.spawnTick += 1

hurdleManager = HurdleManager(3, (30, 90))

```



```

class HurdleClass:
    def __init__(self, x, img, width, height):
        self.x = x
        self.img = img
        self.width = width
        self.height = height
        self.y = ground - height

    def update(self, moveSpeed):
        self.move(moveSpeed)
        self.show()

    def move(self, moveSpeed):
        self.x -= moveSpeed

    def show(self):
        window.blit(self.img, (self.x, self.y))

    def onScreen(self):
        if self.x + self.width > 0:
            return True
        else:
            return False

def mainEventLoop():
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

groundImg = pygame.transform.scale(pygame.image.load('ground.png'), (windowX, int(windowY - ground)))

font1 = pygame.font.Font(('calibri.ttf'), 50)
font2 = pygame.font.Font(('calibri.ttf'), 40)
deathMessage1 = font1.render('You Fell Over!', True, (255, 255, 255))
deathMessage1Shadow = font1.render('You Fell Over!', True, (0, 0, 0))
deathMessage2 = font2.render('Press Space', True, (255, 255, 255))
deathMessage2Shadow = font2.render('Press Space', True, (0, 0, 0))

message1Rect = deathMessage1.get_rect()
message1x = windowX / 2 - message1Rect.width / 2

message2Rect = deathMessage2.get_rect()
message2x = windowX / 2 - message2Rect.width / 2

def showMessage(y):
    window.blit(deathMessage1, (message1x, y))
    window.blit(deathMessage1Shadow, (message1x + 5, y + 5))

    window.blit(deathMessage2, (message2x, y + message1Rect.height))
    window.blit(deathMessage2Shadow, (message2x, message1Rect.height + 5 + y))

score = {'gameScore': 0}

def game():
    player.update()
    while True:
        if player.dead:
            mainEventLoop()
            fall(scoreStr)
        else:
            mainEventLoop()
            window.fill((200, 240, 250))

        player.update()

        hurdleManager.update(True, score['gameScore'] / 50 + 3)
        window.blit(groundImg, (0, ground))

        clock.tick(60)

        scoreStr = font2.render(str(round(score['gameScore'])), True, (0, 0, 0))
        window.blit(scoreStr, (50, 50))

        pygame.display.update()

        score['gameScore'] += 0.1

```

```

def fall(scoreStr):
    space = 0
    while True:
        pressedKeys = pygame.key.get_pressed()

        oldSpace = space
        space = pressedKeys[pygame.K_SPACE]

        mainEventLoop()
        window.fill((200, 240, 250))

        player.update()

        hurdleManager.update(False, score['gameScore'] / 50 + 3)
        window.blit(groundImg, (0, ground))
        clock.tick(60)
        showMessage(50)
        window.blit(scoreStr, (50, 50))
        pygame.display.update()

        spaceEvent = space - oldSpace

    if spaceEvent == 1:
        #Reset Everything

        hurdleManager.hurdleList = []
        player.velocityY = 0
        player.dead = False
        player.y = ground - player.height
        score['gameScore'] = 0

        break
    else:
        mainEventLoop()

game()

```

E. Aset/Gambar



Gambar 1. Duck 1 sebagai gambar gerakan pertama pemain



Gambar 2. Duck 2 sebagai gambar gerakan kedua pemain



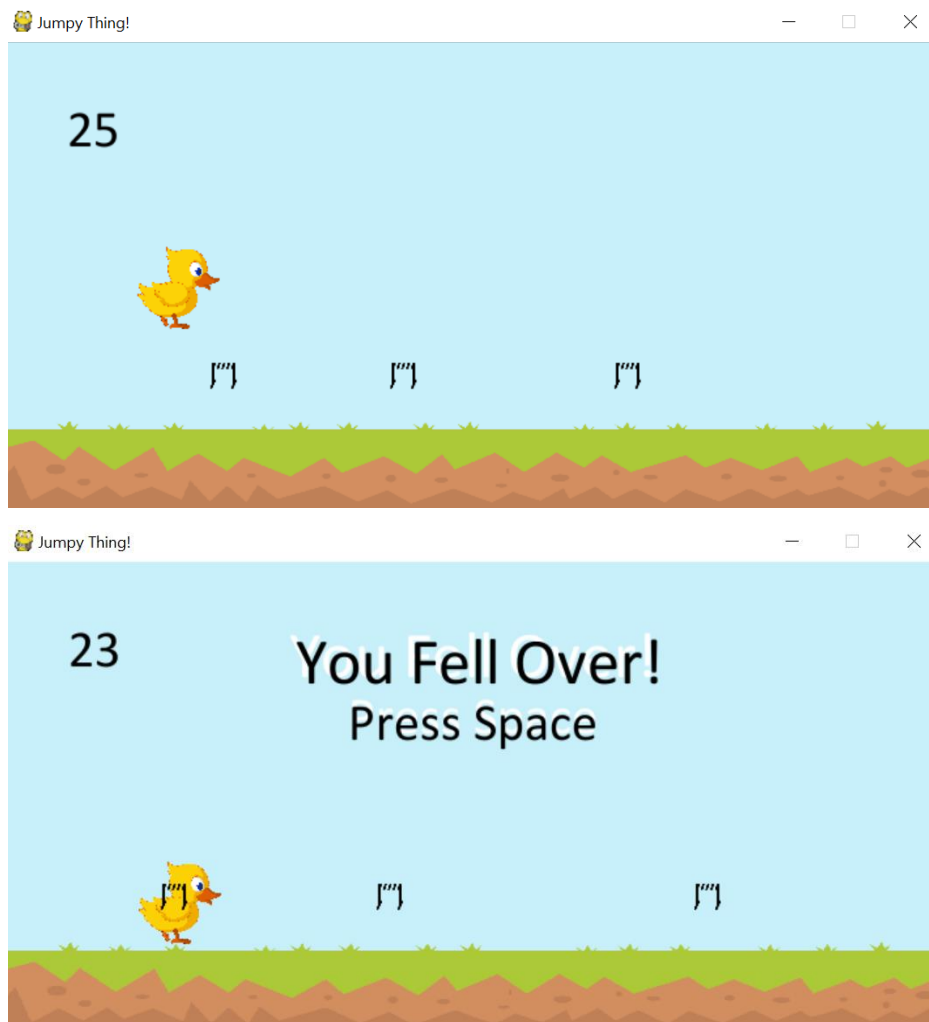
Gambar 3. Ground sebagai tanah ketika pemain berjalan dan pergerakan gawang atau hurdle.

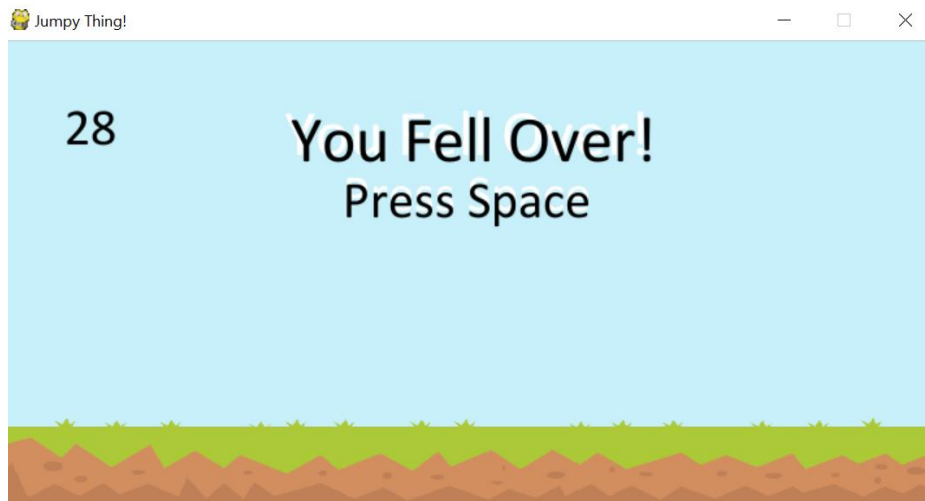


Gambar 4. Gawang atau hurdle yang harus dilewati oleh pemain

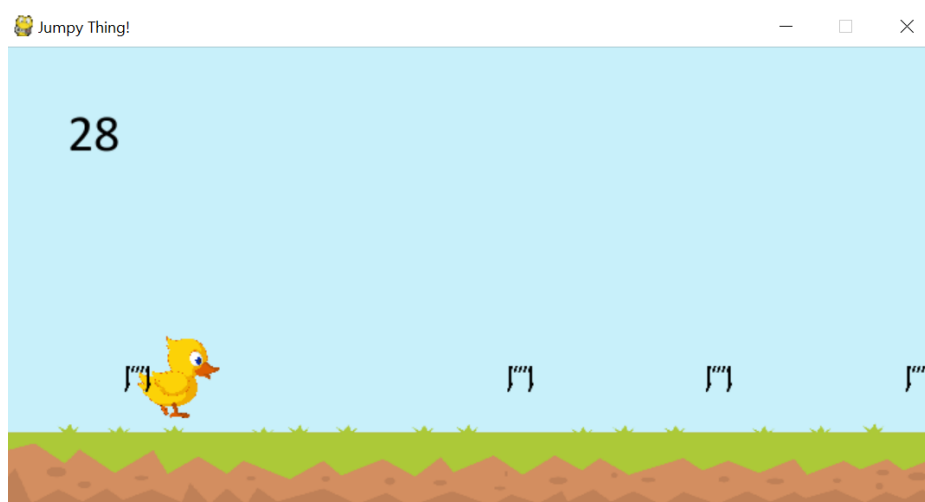
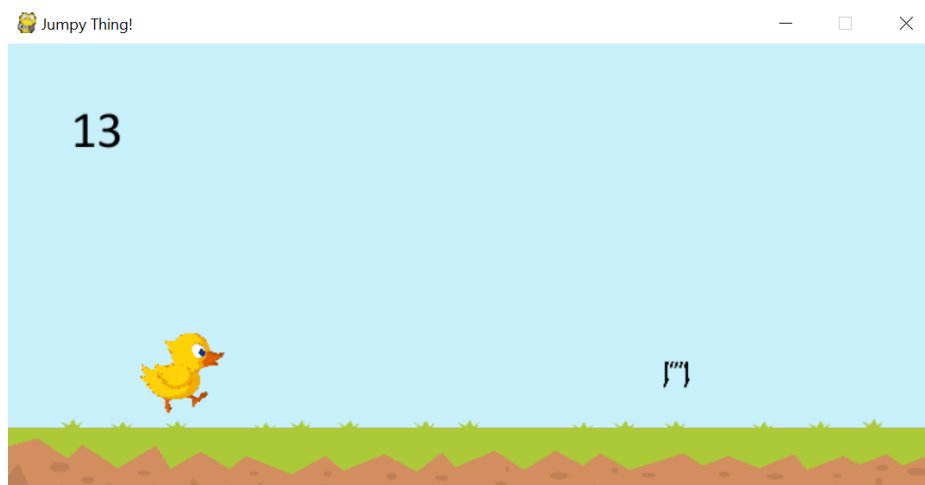
F. Tampilan

- a) Tampilan Awal (game selesai ketika pemain menyentuh satu gawang)





- b) Tampilan Setelah Modifikasi (game selesai ketika pemain menyentuh tiga gawang yang diatur melalui global variabel nyawa)





KIVY

A. Penjelasan

Kivy adalah salah satu program aplikasi yang menggunakan bahasa Python, dan dapat digunakan untuk membuat game dengan platform Android. Kivy ini sendiri merupakan framework yang dibangun menggunakan library dari bahasa pemrograman Python yang bersifat Open Source. Tujuan dikembangkannya framework ini agar dapat membantu developer secara cepat dalam mengembangkan aplikasi yang memiliki tampilan antarmuka inovatif seperti aplikasi yang mendukung multitouch.

B. Cara Instalasi Kivy

Menginstal kivy sebenarnya gampang-gampang susah, caranya sebenarnya sudah ada di dokumentasi resmi kivy <https://kivy.org/#home>. Untuk menggunakan Kivy, Anda perlu Python. Beberapa versi Python dapat diinstal berdampingan, tetapi Kivy harus diinstal untuk setiap versi Python yang ingin Anda gunakan Kivy.

- Catatan:

Untuk Python <3.5 kami menggunakan compiler MinGW. Namun, untuk Python 3.5+ di Windows kami saat ini hanya mendukung compiler MSVC karena masalah Python berikut 4709 tentang MinGW. Umumnya ini seharusnya tidak ada bedanya ketika menggunakan roda dikompilasi.

Setelah python dipasang, buka baris perintah dan pastikan python tersedia dengan mengetik `python --version` pada CMD (Command Prompt) dengan cara menekan tombol Windows lalu ketik CMD atau dengan cara klik start, assesoris, pilih commad prompt.. Kemudian, lakukan hal berikut untuk menginstal.

1. Pastikan Anda memiliki pip dan roda terbaru:

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

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

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

- Catatan:

Jika Anda menemukan MemoryError saat menginstal, tambahkan setelah pip menginstal opsi `--no-cache-dir`.

Untuk Python 3.5+, Anda juga dapat menggunakan backend sudut alih-alih melebur. Ini dapat diinstal dengan:

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

3. Instal kivy:

```
python -m pip instal kivy
```

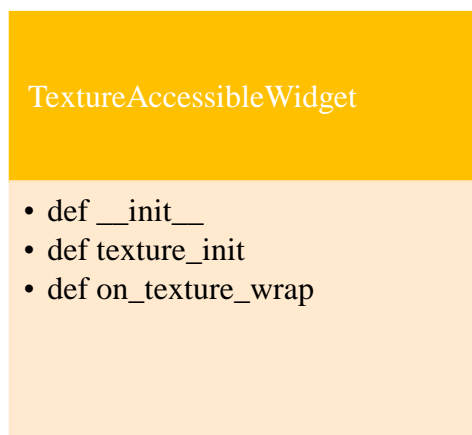
4. (Opsional) Instal contoh kivy:

```
python -m pip instal kivy_examples
```

Jika Anda menemukan izin ditolak kesalahan, coba buka Command prompt sebagai administrator dan coba lagi.

C. Penjelasan OOP Pada Program Pygame

a) Diagram Kelas Pada Program Kivy



b) Penjelasan dan Konsep OOP

- Penjelasan Diagram

- 1) **`def __init__`** pada kelas **`TextureAccessibleWidget`** berfungsi untuk mengorganisasikan serta mendefinisikan kelas pemain.
- 2) **`def texture_init`** pada kelas **`TextureAccessibleWidget`** berfungsi untuk mengatur gambar yang ada pada texture.
- 3) **`def on_texture_wrap`** pada kelas **`TextureAccessibleWidget`** berfungsi untuk mengatur dan mngorganisasikan button yang ada pada texture wrap.

- Kelas dan Objek

Struktur data yang bisa kita gunakan untuk mendefinisikan objek yang menyimpan data bersama-sama nilai-nilai dan perilaku(behavior). Kelas adalah suatu entitas yang merupakan bentuk program dari suatu abstraksi untuk permasalahan dunia

nyata, dan instans dari class yang merupakan realisasi dari beberapa objek. Contoh Kelas dan Objek pada program pygame :

1) Kelas TextureAccessibleWidget

2) texture_init sebagai objek dari kelas TextureAccessibleWidget

- **Override Enheritance**

Method atau properti yang sudah ada dimodifikasi atau diganti nilai properti atau methodnya agar sesuai yang diinginkan. Properti yang digunakan harus sama dengan properti di kelas anak yang dimodifikasi, isi method boleh diganti. Contoh Override Enheritance pada program pygame : -

- **Multiple Enheritance**

Pada Multiple Enheritance subclass dapat mewarisi atribut dan method lebih dari satu parent class. Contoh Multiple Enheritance pada program pygame : -

- **Super Enheritance**

Pada Multiple Enheritance subclass dapat menambah properti dari parent class. Contoh Super Enheritance pada program pygame : -

- **Polymorphism with Function**

Pada Polymorphism with Function, properti di masing-masing class memiliki nama yang sama namun isinya berbeda, dan pemanggilan kelas-kelas tersebut dengan menggunakan dengan menggunakan objek yang terdapat pada global fungsi. Contoh Polymorphism with Function pada program pygame : -

- **Polymorphism with Class**

Pada Polymorphism with Class, metode yang digunakan dengan cara tanpa mengetahui atau tidak peduli apa tipe kelas dari masing-masing objek ini. Pertama-tama dapat membuat perulangan for yang

beriterasi melalui tupel objek. Contoh Polymorphism with Class pada program pygame : -

- Overloading

Overloading merupakan sebuah method dengan multiple cara, dengan beda parameter, dan berbeda dari fungsi yang didefinisikan. Operator overloading merupakan operator yang banyak bentuk. Contoh Overloading pada program pygame : -

- Encapsulation

Enkapsulasi (pembungkusan atau pengkapsulan) merupakan suatu cara untuk menyembunyikan suatu proses atau data didalam sistem aplikasi. Dengan *enkapsulasi*, pengguna dapat memilih *property* dan *method* apa saja yang boleh diakses, dan method yang tidak boleh diakses yaitu dengan menggunakan private method/properti dan public method/properti. Bersifat privat maka tidak dapat diakses di luar kelas dan bersifat public maka dapat diakses diluar kelas. Contoh enkapsulasi pada program pygame : -

- Magic Method

Magic method merupakan operator overloading yang dicapai dengan mendefinisikan metode khusus dalam definisi kelas. Contoh Magic Method pada program pygame : -

D. Program Kivy

a) Program Awal

```
*texture.py - D:\Tugas\Semester 3\PBO\kivy\texture\ORIGINAL\texture.py (3.6.2)*
File Edit Format Run Options Window Help

from kivy.uix.widget import Widget
from kivy.properties import ObjectProperty, ListProperty, StringProperty
from kivy.lang import Builder
from kivy.clock import Clock
from kivy.base import runTouchApp

class TextureAccessibleWidget(Widget):
    texture = ObjectProperty(None)
    tex_coords = ListProperty([0, 0, 1, 0, 1, 1, 0, 1])
    texture_wrap = StringProperty('clamp_to_edge')

    def __init__(self, **kwargs):
        super(TextureAccessibleWidget, self).__init__(**kwargs)
        Clock.schedule_once(self.texture_init, 0)

    def texture_init(self, *args):
        self.texture = self.canvas.children[-1].texture

    def on_texture_wrap(self, instance, value):
        self.texture.wrap = value

root = Builder.load_string('''
<TextureAccessibleWidget>:
    canvas:
        Rectangle:
            pos: self.pos
            size: self.size
            source: 'texture_example_image.png'
            tex_coords: root.tex_coords

<SliderWithValue@BoxLayout>:
    min: 0.0
    max: 1.0
    value: slider.value
    Slider:
        id: slider
        orientation: root.orientation
        min: root.min
        max: root.max
        value: 1.0
    Label:
        size_hint: None, None
        size: min(root.size), min(root.size)
        text: str(slider.value)[:4]
''')
```

```

BoxLayout:
    orientation: 'vertical'
    BoxLayout:
        SliderWithValue:
            orientation: 'vertical'
            size_hint_x: None
            width: dp(40)
            min: 0
            max: 5
            value: 1
            on_value: taw.tex_coords[5] = self.value
            on_value: taw.tex_coords[7] = self.value
        SliderWithValue:
            orientation: 'vertical'
            size_hint_x: None
            width: dp(40)
            min: 0
            max: taw_container.height
            value: 0.5*taw_container.height
            on_value: taw.height = self.value
        AnchorLayout:
            id: taw_container
            anchor_x: 'left'
            anchor_y: 'bottom'
            TextureAccessibleWidget:
                id: taw
                size_hint: None, None
BoxLayout:
    size_hint_y: None
    height: dp(80)
    BoxLayout:
        orientation: 'vertical'
        size_hint_x: None
        width: dp(80)
        Label:
            text: 'size'
            text_size: self.size
            halign: 'right'
            valign: 'middle'
        Label:
            text: 'tex_coords'
            text_size: self.size
            halign: 'left'
            valign: 'middle'
    BoxLayout:
        orientation: 'vertical'
        SliderWithValue:
            min: 0
            max: taw_container.width
            value: 0.5*taw_container.width
            on_value: taw.width = self.value

```

```

        SliderWithValue:
            min: 0.
            max: 5.
            value: 1.
            on_value: taw.tex_coords[2] = self.value
            on_value: taw.tex_coords[4] = self.value

BoxLayout:
    size_hint_y: None
    height: dp(50)
    Label:
        text: 'texture wrap:'
        text_size: self.size
        valign: 'middle'
        halign: 'center'
    Button:
        text: 'clamp_to_edge'
        on_press: taw.texture_wrap = 'clamp_to_edge'
    Button:
        text: 'repeat'
        on_press: taw.texture_wrap = 'repeat'
    Button:
        text: 'mirrored_repeat'
        on_press: taw.texture_wrap = 'mirrored_repeat'
'''

runTouchApp(root)

```

b) Program Setelah Modifikasi

texture_2.py - D:\Tugas\Semester 3\PBO\kivy\texture\texture_2.py (3.6.2)

File Edit Format Run Options Window Help

```

from kivy.uix.widget import Widget
from kivy.app import App
from kivy.properties import ObjectProperty, ListProperty, StringProperty
from kivy.lang import Builder
from kivy.clock import Clock
from kivy.base import runTouchApp

class TextureAccessibleWidget(Widget):
    texture = ObjectProperty(None)
    tex_coords = ListProperty([0, 0, 1, 0, 1, 1, 0, 1])
    texture_wrap = StringProperty('clamp_to_edge')

    def __init__(self, **kwargs):
        super(TextureAccessibleWidget, self).__init__(**kwargs)
        Clock.schedule_once(self.texture_init, 0)

    def texture_init(self, *args):
        self.texture = self.canvas.children[-1].texture

    def on_texture_wrap(self, instance, value):
        self.texture.wrap = value

```

```

root = Builder.load_string('''
<TextureAccessibleWidget>:
    canvas:
        Rectangle:
            pos: self.pos
            size: self.size
            source: 'texture_example_image.png'
            tex_coords: root.tex_coords

<SliderWithValue@BoxLayout>:
    min: 0.0
    max: 1.0
    value: slider.value
    Slider:
        id: slider
        orientation: root.orientation
        min: root.min
        max: root.max
        value: 1.0
    Label:
        size_hint: None, None
        size: min(root.size), min(root.size)
        text: str(slider.value)[:4]

BoxLayout:
    orientation: 'vertical'
    BoxLayout:
        SliderWithValue:
            orientation: 'vertical'
            size_hint_x: None
            width: dp(40)
            min: 0
            max: 5
            value: 1
            on_value: taw.tex_coords[5] = self.value
            on_value: taw.tex_coords[7] = self.value

        SliderWithValue:
            orientation: 'vertical'
            size_hint_x: None
            width: dp(40)
            min: 0
            max: taw_container.height
            value: 0.5*taw_container.height
            on_value: taw.height = self.value
    AnchorLayout:
        id: taw_container
        anchor_x: 'center'
        anchor_y: 'center'
        TextureAccessibleWidget:
            id: taw
            size_hint: None, None

BoxLayout:
    size_hint_y: None
    height: dp(80)
    BoxLayout:
        orientation: 'vertical'
        size_hint_x: None
        width: dp(80)

```

```

        Label:
            text: 'size'
            text_size: self.size
            halign: 'right'
            valign: 'middle'
        Label:
            text: 'tex_coords'
            text_size: self.size
            halign: 'right'
            valign: 'middle'
    BoxLayout:
        orientation: 'vertical'
        SliderWithValue:
            min: 0
            max: taw_container.width
            value: 0.5*taw_container.width
            on_value: taw.width = self.value
        SliderWithValue:
            min: 0.
            max: 5.
            value: 1.
            on_value: taw.tex_coords[2] = self.value
            on_value: taw.tex_coords[4] = self.value
BoxLayout:
    size_hint_y: None
    height: dp(80)
    BoxLayout:
        orientation: 'vertical'
        size_hint_x: None
        width: dp(80)
        Label:
            text: 'size All'
            text_size: self.size
            halign: 'right'
            valign: 'middle'
        Label:
            text: 'tex_coords All'
            text_size: self.size
            halign: 'right'
            valign: 'middle'
    BoxLayout :
        orientation: 'vertical'
        SliderWithValue:
            min: 0
            max: (taw_container.width+taw_container.height)
            value: (0.1*taw_container.width)+(0.1*taw_container.height)
            on_value: taw.width = self.value
            on_value: taw.height = self.value
        SliderWithValue:
            min: 0
            max: 5
            value: 1
            on_value: taw.tex_coords[2] = self.value
            on_value: taw.tex_coords[4] = self.value
            on_value: taw.tex_coords[5] = self.value
            on_value: taw.tex_coords[7] = self.value

```

```

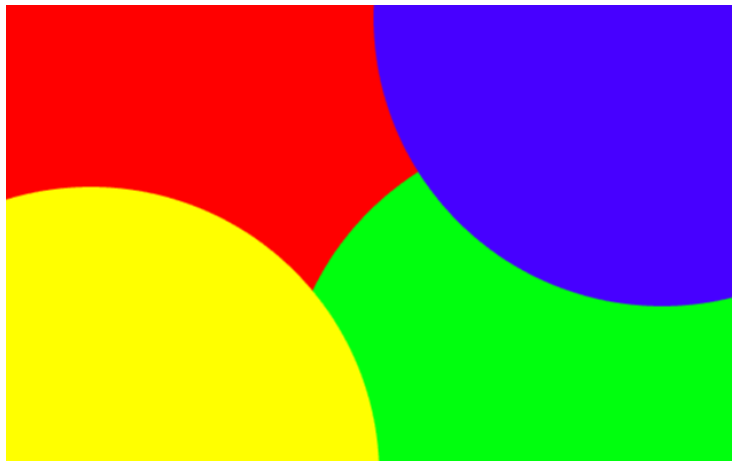
BoxLayout:
    size_hint_y: None
    height: dp(50)
    Label:
        text: 'texture wrap:'
        text_size: self.size
        valign: 'middle'
        halign: 'center'
    Button:
        text: 'repeat'
        on_press: taw.texture_wrap = 'repeat'
    Button:
        text: 'mirrored_repeat'
        on_press: taw.texture_wrap = 'mirrored_repeat'

'''

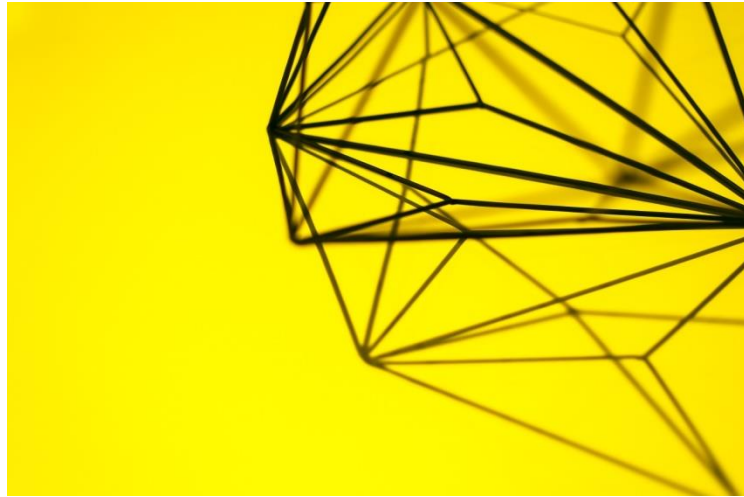
runTouchApp(root)

```

E. Aset/Gambar



Gambar 1. Contoh gambar untuk Aplikasi Texture (Sebelum Modifikasi)



Gambar 2. Contoh gambar untuk Aplikasi Texture (Sesudah Modifikasi)

F. Tampilan

- Tampilan Awal



- Tampilan Setelah Modifikasi

