TUGAS BESAR GRAFIKA KOMPUTER
Pengembangan *Game* 'JUMP O1' Menggunakan Bahasa Pemrograman Python Berbasis Desktop

Diusulkan oleh :

Nama	: Ferza Reyaldi
NIM	: 09021281924060
Program Studi	: Teknik Informatika
Kelas	: 4 REG A

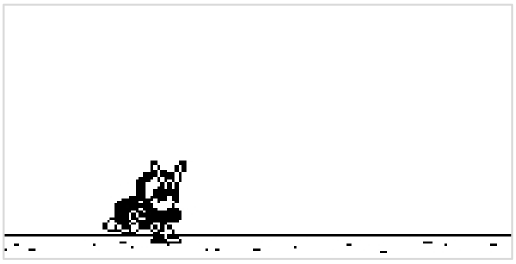
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SRIWIJAYA
2021

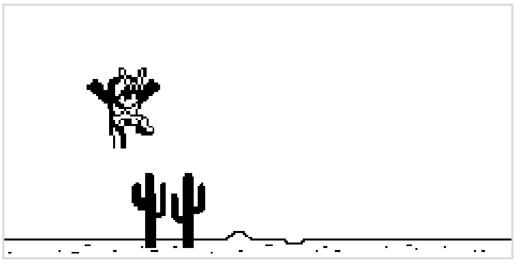
NAMA PROJECT

Nama *project* yang dikembangkan adalah JUMP O1! (*Jump Zero-One!*). Game ini merupakan game dengan sentuhan grafis *retro* (biasa disebut game 8-bit), dengan genre game platform (game lompat-lompat).

STORY BOARD

Tujuan dari game adalah meraih skor setinggi mungkin dengan cara terus berlari dan lompat untuk menghindari *obstacle*(rintangan) yang ada. Semakin tinggi skor, semakin cepat dan banyak *obstacle* yang diberikan.


Gambar 1 Karakter berlari


Gambar 2 Karakter melompat menghindari *obstacle*

ASET YANG DIBUTUHKAN

Aset yang dibutuhkan untuk membangun permainan JUMP O1!:

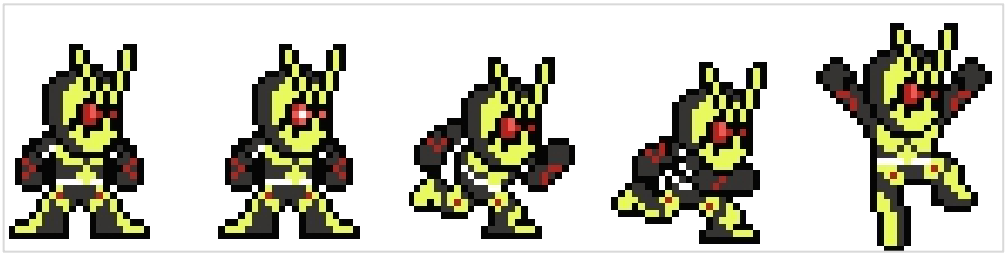
- Text-Editor, seperti: VS Code.
- Instalasi bahasa Python.
- Beberapa library Python: Pygame.
- 7 aset gambar
- 5 aset suara.
- Aplikasi desain gambar: Figma.
- Font-family: Sec Zero One.

DOKUMENTASI (USER-GUIDE)

Berikut dokumentasi dan tuntunan untuk pengguna (*user-guide*) permainan JUMP O1:

1. **Character**

   Karakter yang dimainkan di game JUMP O1 berdasarkan pemeran series superhero fiktif dari Jepanh, yaitu Kamen Rider Zero-One, yang didesain dengan gaya 8-bit (*retro*)


Gambar 3 Desain Karakter Zero-One pada game JUMP O1

2. **Obstacle**

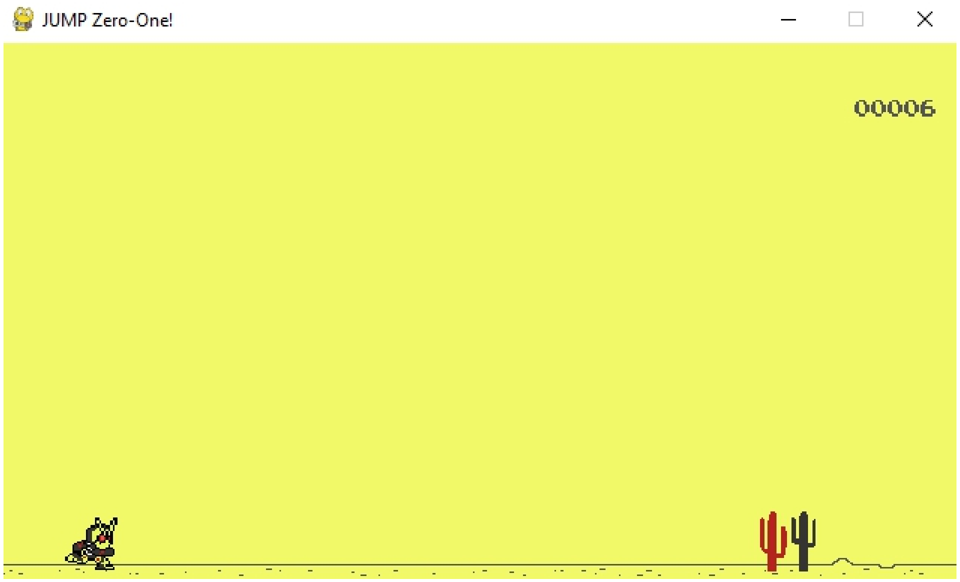
   Obstacle atau rintangan berupa kaktus berwarna merah dan hitam, terinspirasi dari warna desain Zero-One

Gambar 4 Desain *Obstacle*

3. **Gameplay (User-Guide)**

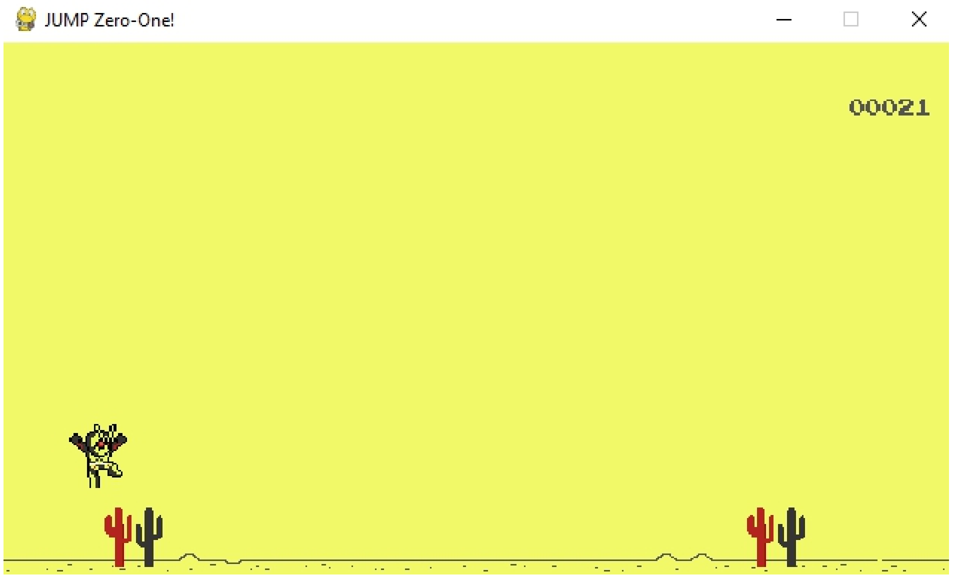
   Cara memainkan permainan JUMP O1! sangat sederhana. Game ini hanya menggunakan satu tombol,yaitu **Space** untuk melompat.

   - Game dimulai dengan menekan tombol **Space**.
   - Zero-One terus berlari secara otomatis.



Gambar 5 Zero-One berlari

- Zero-One melompat setelah ditekan tombol **Space.**



Gambar 6 Zero-One melompat.

- Semakin jauh Zero-One berlari dan menghindari *obstacle*, score semakin tinggi dan semakin cepat dan banyak rintangan yang diberikan.

PROGRESS REPORT

1. **Week-1 (21-27 Januari 2021)**
   - Menentukan bahasa pemrograman yang digunakan: Python
   - Menentukan *tools* yang digunakan: VS Code
   - Menginstal Library PyGame sebagai alat bantu dalam membuat game.
   - Menentukan genre Game yang ingin dibuat: Game Platformer (terinspirasi dari Dino Run di Browser).

2. **Week-2 (28 Januari - 3 Februari 2021)**
   - Memiliki gambaran *gameplay* yang diusung.
   - Menentukan gaya desain game: Retro (biasa disebut 8-bit).
   - Menyelesaikan desain *sprite* karakter game: Kamen Rider Zero-One.
   - Telah memilki background game.
   - Telah membuat nama dan logo game.

3. **Week-3 (4-10 Februari 2021)**
   - Menyelesaikan desain *obstacle* (rintangan).
   - Menonton beberapa tutorial video pengembangan game menggunakan bahasa pemrograman python.

4. **Week-4 (11-17 Februari 2021)**
   - Membuat desain tombol *replay*.

- Menentukan font teks dalam game.
- Melanjutkan menonton beberapa tutorial video pengembangan game menggunakan bahasa pemrograman python.

5. **Week-5 (18-24 Februari 2021)**
   - Membuat desain teks pada game seperti teks *Game Over* dan *score*.
   - Melanjutkan menonton beberapa tutorial video pengembangan game menggunakan bahasa pemrograman python.

6. **Week-6 (25 Februari - 3 Maret 2021)**
   - Membuat storyboard.
   - Melanjutkan menonton beberapa tutorial video pengembangan game menggunakan bahasa pemrograman python.

7. **Week-7 (4-10 Maret 2021)**
   - Mengumpul aset suara game.
   - Melanjutkan beberapa tutorial video pengembangan game menggunakan bahasa pemrograman python.

8. **Week-8 (11-17 Maret 2021)**
   - Memulai penulisan code dengan menerapkan konsep OOP (*Object-Oriented Programming*). Code yang ditulis adalah *blueprint* objek yang ingin dibuat: kelas ZeroOne (karakter) dan kelas Ground (Latar tempat).

9. **Week-9 (18-24 Maret 2021)**
   - Melanjutkan penulisan code. Code yang ditulis adalah *blueprint* objek yang ingin dibuat: kelas Cactus (*obstacle*) dan kelas ScoreBoard.

10. **Week-10 (25-31 Maret 2021)**
    - Melanjutkan penulisan code. Memulai menuliskan *method* untuk menjalankan game, yaitu: load_image(), load_sprite_sheet(), game_over_display_message(), extractDigits().

11. **Week-11 (1-7 April 2021)**
    - Meelanjutkan penulisan code. Memulai menuliskan *method* untuk menjalankan game, yaitu: introduction_screen(), gameplay() dan main().

12. **Week-12 (8-14 April 2021)**
    - Mencoba menjalankan code yang dibuat.
    - Mengganti warna latar game, dari abu-abu menjadi kuning.
    - Melakukan debugging.

13. **Week-13 (15-21 April 2021)**
    - Memperbaiki code yang belum baik.
    - Melakukan debugging terhadap bug yang tersisa.
    - Mulai menerapkan prinsip clean code.

14. **Week-14 (22-28 April 2021)**
    - Menyusun dokumentasi.
    - Membuat video demo game.

SOURCE CODE

```python
#Developer: Ferza Reyaldi 09021281924060


import os, sys, pygame, random

from pygame import *


pygame.init()


screen_size_display = (width_screen, height_screen) = (640, 360)

FPS = 60

gravity = 0.6


black_color = (0,0,0)

white_color = (255,255,255)

bg_color = (242,249,104)


highest_scores = 0


screen_layout_display = pygame.display.set_mode(screen_size_display)

time_clock = pygame.time.Clock()

pygame.display.set_caption("JUMP Zero-One!")
```

```
#Load and Play Music
pygame.mixer.music.load('suaras/Theme.wav')
pygame.mixer.music.play(-1, 0.0)
pygame.mixer.music.set_volume(0.25)

jump_sound = pygame.mixer.Sound('suaras/jump.wav')
die_sound = pygame.mixer.Sound('suaras/die.wav')
checkPoint_sound = pygame.mixer.Sound('suaras/checkPoint.wav')

def load_image(name, sx=-1, sy=-1, colorkey=None,):
    fullname = os.path.join('images', name)
    img = pygame.image.load(fullname)
    img = img.convert()
    if colorkey is not None:
        if colorkey == -1:
            colorkey = img.get_at((0, 0))
        img.set_colorkey(colorkey, RLEACCEL)

    if sx != -1 or sy != -1:
        img = pygame.transform.scale(img, (sx, sy))

    return (img, img.get_rect())

def load_sprite_sheet(s_name, namex, namey, scx = -1, scy = -
1, c_key = None,):
    fullname = os.path.join('images', s_name)
    sh = pygame.image.load(fullname)
    sh = sh.convert()


    sh_rect = sh.get_rect()


    sprites = []


    sx = sh_rect.width/namex

    sy = sh_rect.height/namey


    for i in range(0, namey):

        for j in range(0, namex):

            rect = pygame.Rect((j*sx,i*sy,sx,sy))

            img = pygame.Surface(rect.size)

            img = img.convert()

            img.blit(sh,(0,0),rect)


            if c_key is not None:

                if c_key == -1:

                    c_key = img.get_at((0, 0))

                img.set_colorkey(c_key, RLEACCEL)


            if scx != -1 or scy != -1:

                img = pygame.transform.scale(img, (scx, scy))


            sprites.append(img)


    sprite_rect = sprites[0].get_rect()
```

```
    return sprites,sprite_rect


def gameover_display_message(rbtn_image, gmo_image):
    rbtn_rect = rbtn_image.get_rect()
    rbtn_rect.centerx = width_screen / 2
    rbtn_rect.top = height_screen * 0.52

    gmo_rect = gmo_image.get_rect()
    gmo_rect.centerx = width_screen / 2
    gmo_rect.centery = height_screen * 0.35

    screen_layout_display.blit(rbtn_image, rbtn_rect)
    screen_layout_display.blit(gmo_image, gmo_rect)

def extractDigits(num):
    if num > -1:
        d = []
        i = 0
        while(num / 10 != 0):
            d.append(num % 10)
            num = int(num / 10)

        d.append(num % 10)
        for i in range(len(d),5):
            d.append(0)
        d.reverse()
        return d


class ZeroOne():
    def __init__(self, sx=-1, sy=-1):
        self.imgs, self.rect = load_sprite_sheet('01.png', 5, 1, s
x, sy, -1)
        self.rect.bottom = int(0.98 * height_screen)
        self.rect.left = width_screen / 15
        self.image = self.imgs[0]
        self.index = 0
        self.counter = 0
        self.score = 0
        self.jumping = False
        self.dead = False
        self.running = True
        self.blinking = False
        self.movement = [0,0]
        self.jumpSpeed = 11.5

        self.stand_position_width = self.rect.width

    def draw(self):
        screen_layout_display.blit(self.image, self.rect)

    def checkbounds(self):
        if self.rect.bottom > int(0.98 * height_screen):
            self.rect.bottom = int(0.98 * height_screen)
            self.jumping = False
```

```
def update(self):
    if self.jumping:
        self.movement[1] = self.movement[1] + gravity

    if self.jumping:
        self.index = 4
    elif self.blinking:
        if self.index == 0:
            if self.counter % 400 == 399:
                self.index = (self.index + 1)%2
        else:
            if self.counter % 20 == 19:
                self.index = (self.index + 1)%2

    elif not self.running:
        if self.counter % 5 == 0:
            self.index = (self.index + 1)%2
    else:
        if self.counter % 5 == 0:
            self.index = (self.index + 1)%2 + 2

    if self.dead:
       self.index = 4

    if self.running:
        self.image = self.imgs[self.index]
```

```
            self.rect.width = self.stand_position_width

        self.rect = self.rect.move(self.movement)
        self.checkbounds()

        if not self.dead and self.counter % 7 == 6 and self.blinki
ng == False:
            self.score += 1
            if self.score % 100 == 0 and self.score != 0:
                if pygame.mixer.get_init() != None:
                    checkPoint_sound.play()

        self.counter = (self.counter + 1)

class Cactus(pygame.sprite.Sprite):
    def __init__(self, speed=5, sx=-1, sy=-1):
        pygame.sprite.Sprite.__init__(self,self.containers)
        self.imgs, self.rect = load_sprite_sheet('obstacle.png', 3
, 1, sx, sy, -1)
        self.rect.bottom = int(0.98 * height_screen)
        self.rect.left = width_screen + self.rect.width
        self.image = self.imgs[random.randrange(0, 3)]
        self.movement = [-1*speed,0]

    def draw(self):
        screen_layout_display.blit(self.image, self.rect)

    def update(self):
        self.rect = self.rect.move(self.movement)


        if self.rect.right < 0:
            self.kill()


class Ground():
    def __init__(self,speed=-5):
        self.image,self.rect = load_image('ground.png',-1,-1,-1)
        self.image1,self.rect1 = load_image('ground.png',-1,-1,-1)
        self.rect.bottom = height_screen
        self.rect1.bottom = height_screen
        self.rect1.left = self.rect.right
        self.speed = speed


    def draw(self):
        screen_layout_display.blit(self.image, self.rect)
        screen_layout_display.blit(self.image1, self.rect1)


    def update(self):
        self.rect.left += self.speed
        self.rect1.left += self.speed


        if self.rect.right < 0:
            self.rect.left = self.rect1.right


        if self.rect1.right < 0:
            self.rect1.left = self.rect.right
```

```
class Scoreboard():

    def __init__(self,x=-1,y=-1):

        self.score = 0

        self.scre_img, self.screrect = load_sprite_sheet('numbers.
png', 12, 1, 11, int(11 * 6 / 5), -1)

        self.image = pygame.Surface((55,int(11*6/5)))

        self.rect = self.image.get_rect()

        if x == -1:

            self.rect.left = width_screen * 0.89

        else:

            self.rect.left = x

        if y == -1:

            self.rect.top = height_screen * 0.1

        else:

            self.rect.top = y


    def draw(self):

        screen_layout_display.blit(self.image, self.rect)


    def update(self,score):

        score_digits = extractDigits(score)

        self.image.fill((bg_color))

        for s in score_digits:

            self.image.blit(self.scre_img[s], self.screrect)

            self.screrect.left += self.screrect.width

        self.screrect.left = 0
```

```
def introduction_screen():
    ado_zeroone = ZeroOne(44,47)
    ado_zeroone.blinking = True
    starting_game = False

    t_ground,t_ground_rect = load_sprite_sheet('ground.png',1,1,-1,-1,-1)
    t_ground_rect.left = 0
    t_ground_rect.bottom = height_screen

    logo,l_rect = load_image('Logo.png',311,100,-1)
    l_rect.centerx = width_screen * 0.6
    l_rect.centery = height_screen * 0.6
    while not starting_game:
        if pygame.display.get_surface() == None:
            print("Couldn't load display surface")
            return True
        else:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    return True
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_SPACE or event.key == pygame.K_UP:
                        ado_zeroone.jumping = True
                        ado_zeroone.blinking = False
                        ado_zeroone.movement[1] = -1*ado_zeroone.jumpSpeed
```

```python
        ado_zeroone.update()

        if pygame.display.get_surface() != None:
            screen_layout_display.fill(bg_color)
            screen_layout_display.blit(t_ground[0], t_ground_rect)
            if ado_zeroone.blinking:
                screen_layout_display.blit(logo, l_rect)
            ado_zeroone.draw()

            pygame.display.update()

        time_clock.tick(FPS)
        if ado_zeroone.jumping == False and ado_zeroone.blinking == False:
            starting_game = True

def gameplay():
    global highest_scores
    gp = 4
    s_Menu = False
    g_Over = False
    g_exit = False
    gamer_zeroone = ZeroOne(44,47)
    new_grnd = Ground(-1*gp)
    score_boards = Scoreboard()
    highScore = Scoreboard(width_screen * 0.78)
    counter = 0
    cactusan = pygame.sprite.Group()

    last_end_obs = pygame.sprite.Group()


    Cactus.containers = cactusan


    rbtn_image,rbtn_rect = load_image('replay.png',50,50,-1)

    gmo_image,gmo_rect = load_image('GO.png',332,34,-1)


    t_images,t_rect = load_sprite_sheet('numbers.png',12,1,11,int(
11*6/5),-1)

    ado_image = pygame.Surface((22,int(11*6/5)))

    ado_rect = ado_image.get_rect()

    ado_image.fill(bg_color)

    ado_image.blit(t_images[10],t_rect)

    t_rect.left += t_rect.width

    ado_image.blit(t_images[11],t_rect)

    ado_rect.top = height_screen * 0.1

    ado_rect.left = width_screen * 0.73


    while not g_exit:

        while s_Menu:

            pass

        while not g_Over:

            if pygame.display.get_surface() == None:

                print("Couldn't load display surface")

                g_exit = True

                g_Over = True
            else:
                for event in pygame.event.get():
                    if event.type == pygame.QUIT:
                        g_exit = True
                        g_Over = True

                    if event.type == pygame.KEYDOWN:
                        if event.key == pygame.K_SPACE or event.ke
y == pygame.K_UP:
                            if gamer_zeroone.rect.bottom == int(0.
98 * height_screen):
                                gamer_zeroone.jumping = True
                                if pygame.mixer.get_init() != None
:
                                    jump_sound.play()
                                gamer_zeroone.movement[1] = -
1*gamer_zeroone.jumpSpeed

            for c in cactusan:
                c.movement[0] = -1*gp
                if pygame.sprite.collide_mask(gamer_zeroone,c):
                    gamer_zeroone.dead = True
                    if pygame.mixer.get_init() != None:
                        die_sound.play()

            if len(cactusan) < 2:
                if len(cactusan) == 0:
                    last_end_obs.empty()
                    last_end_obs.add(Cactus(gp,40,40))
                else:
                    for l in last_end_obs:
                        if l.rect.right < width_screen*0.7 and ran
dom.randrange(0, 50) == 10:
                            last_end_obs.empty()
                            last_end_obs.add(Cactus(gp, 40, 40))

            gamer_zeroone.update()
            cactusan.update()
            new_grnd.update()
            score_boards.update(gamer_zeroone.score)
            highScore.update(highest_scores)

            if pygame.display.get_surface() != None:
                screen_layout_display.fill(bg_color)
                new_grnd.draw()
                score_boards.draw()
                if highest_scores != 0:
                    highScore.draw()
                    screen_layout_display.blit(ado_image, ado_rect
)
                cactusan.draw(screen_layout_display)
                gamer_zeroone.draw()

                pygame.display.update()
            time_clock.tick(FPS)

            if gamer_zeroone.dead:
```

```
            g_Over = True
            if gamer_zeroone.score > highest_scores:
                highest_scores = gamer_zeroone.score

        if counter%700 == 699:
            new_grnd.speed -= 1
            gp += 1

        counter = (counter + 1)

    if g_exit:
        break

    while g_Over:
        if pygame.display.get_surface() == None:
            print("Couldn't load display surface")
            g_exit = True
            g_Over = False
        else:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    g_exit = True
                    g_Over = False
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_ESCAPE:
                        g_exit = True
                        g_Over = False
                        if event.key == pygame.K_RETURN or event.k
ey == pygame.K_SPACE:
                            g_Over = False
                            gameplay()
            highScore.update(highest_scores)
            if pygame.display.get_surface() != None:
                gameover_display_message(rbtn_image, gmo_image)
                if highest_scores != 0:
                    highScore.draw()
                    screen_layout_display.blit(ado_image, ado_rect
)
                pygame.display.update()
            time_clock.tick(FPS)

    pygame.quit()
    quit()

def main():
    isGameQuit = introduction_screen()
    if not isGameQuit:
        gameplay()

main()
```