Project Report – Steam Man

13006107 Introduction to Computers and Programming

Software Engineering Program

Faculty of Engineering, KMITL

By

63011193 Naral Chalermchaikosol

Steam Man

Introduction

Steam Man is a 2D side scroller running game.

You gain points by jumping over obstacles.

Do your best to avoid them, as bumping into them results in game over.

LIBRARIES USED:

- **PYGAME** – Driver library of the project. This allows for sprite animation, background music and more.
- **RANDOM** – Randomize the height of each pipe and each raven.
- **SYS** – Allows for pygame application exit.

MOTIVATION / INSPIRATION:

8-bit side scroller games used to dominate the market when I was young, and I was a huge fan of those. Whenever I think of creating a video game, these games come to mind. Creating "Steam Man" not only helped me bring back good memories but also refined my coding skills.

SCREENSHOTS

STARTING SCREEN / GAME OVER SCREEN

GAMEPLAY MECHANICS

The game features ravens and pipes as obstacles,

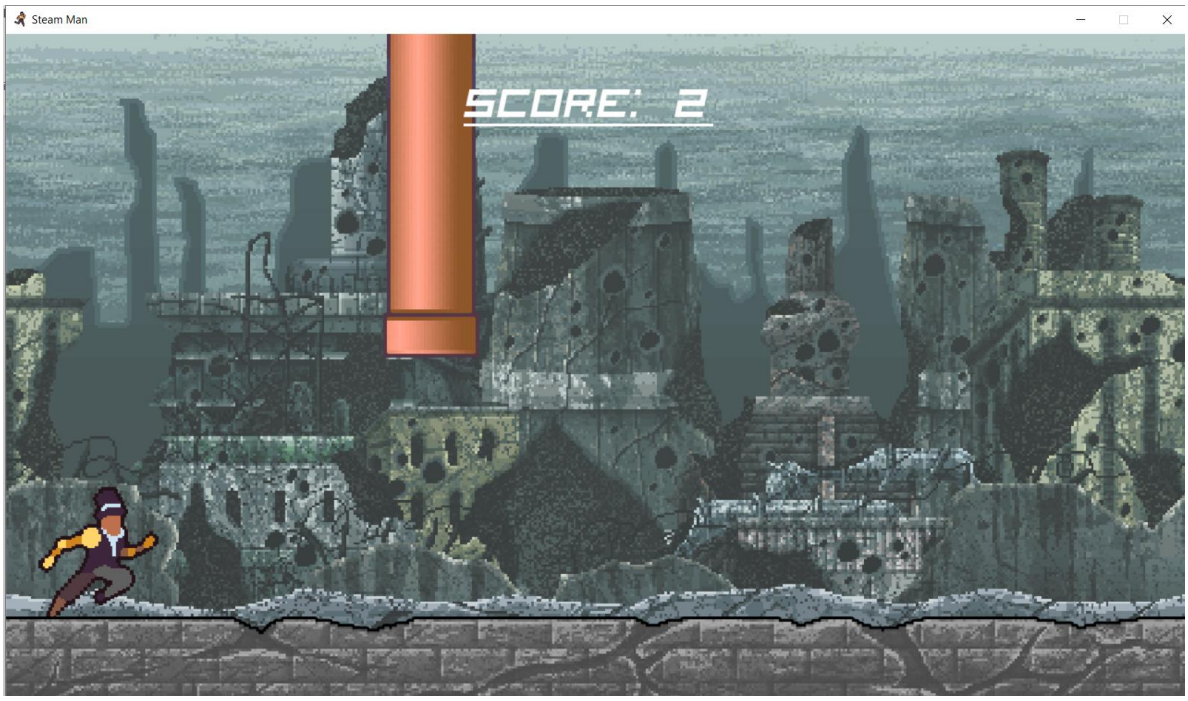
bumping into them results in game over.

However, successfully jumping over a pipe grants you a point.

Your character can jump up to three times before needing to land.

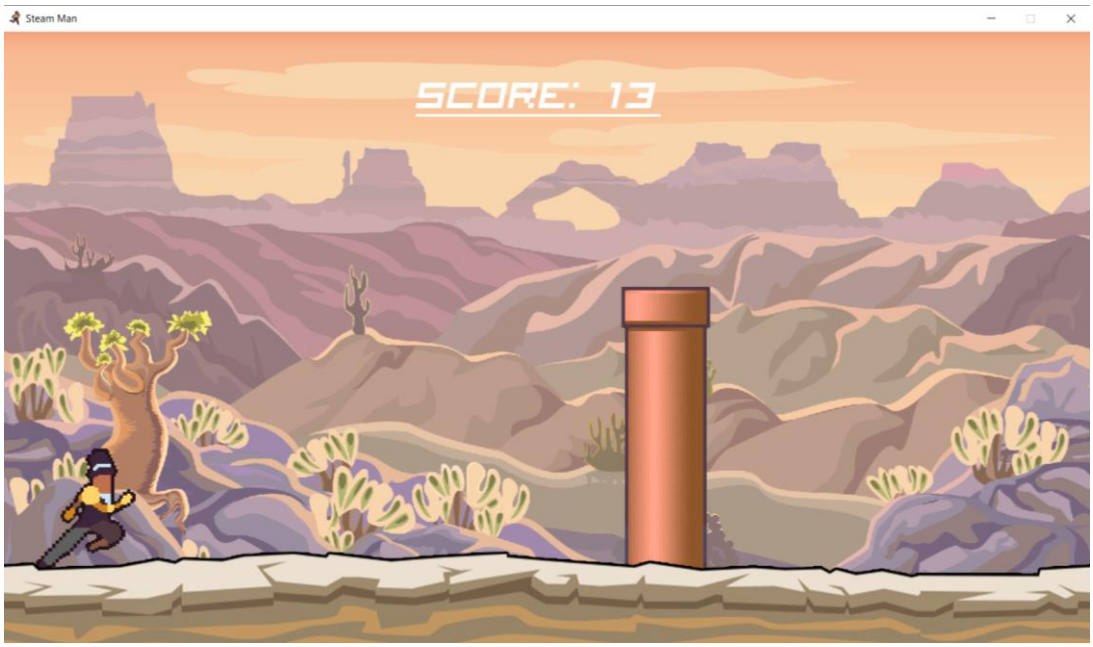
DIFFERENT SCENERIES WITH DIFFERENT BACKGROUND MUSIC

Swamp



Destroyed City

Snow land



Desert

SOURCE CODE PREVIEW

Note – the .py file is attached to the zip

```
import pygame, sys, random


def draw_floor():
    screen.blit(floor_surface, (floor_x_pos, 0))
    screen.blit(floor_surface, (floor_x_pos + 1280, 0))


def create_bird():
    random_bird_pos = random.choice(bird_altitude)
    bird = bird_surface.get_rect(midtop=(1300, random_bird_pos))
    return bird


def create_pipe():
    random_pipe_pos = random.choice(pipe_height)
    bottom_pipe = pipe_surface.get_rect(midtop=(1300, random_pipe_pos))
    top_pipe = pipe_surface.get_rect(midbottom=(1300, random_pipe_pos - 500))
    return bottom_pipe, top_pipe


def move_birds(birds):
    for bird in birds:
        bird.centerx -= 8 * speed_factor
    visible_birds = [bird for bird in birds if bird.right > -50]
    return visible_birds


def move_pipes(pipes):
    for pipe in pipes:
        pipe.centerx -= 5 * speed_factor
    visible_pipes = [pipe for pipe in pipes if pipe.right > -50]
    return visible_pipes


def draw_birds(birds):
    for bird in birds:
        screen.blit(bird_surface, bird)


def draw_pipes(pipes):
    for pipe in pipes:
        if pipe.bottom >= 720:
            screen.blit(pipe_surface, pipe)
        else:
            flip_pipe = pygame.transform.flip(pipe_surface, False, True)
            screen.blit(flip_pipe, pipe)


def check_collision_bird(birds):
    global death
    for bird in birds:
        if mc_rect.colliderect(bird):
            death = True
            death_sound.play()
            # pygame.mixer.music.fadeout(750)
            return False
    return True
```

```
def check_collision(pipes):
    global death
    for pipe in pipes:
        if mc_rect.colliderect(pipe):
            death = True
            death_sound.play()
            # pygame.mixer.music.fadeout(750)
            return False
    return True


def bg_and_music_change():
    if bg_index == 0:
        pygame.mixer.music.load("assets/sfx/bg_music_swamp.mp3")
    elif bg_index == 1:
        pygame.mixer.music.load("assets/sfx/bg_music_war.mp3")
    elif bg_index == 2:
        pygame.mixer.music.load("assets/sfx/bg_music_snow.mp3")
    else:
        pygame.mixer.music.load("assets/sfx/bg_music_desert.mp3")
    pygame.mixer.music.play(-1)
    new_bg = bg_frames[bg_index]
    new_bg = pygame.transform.scale(new_bg, (1280, 720))
    return new_bg


def floor_change():
    new_floor = floor_frames[floor_index]
    new_floor = pygame.transform.scale(new_floor, (1280, 720))
    return new_floor


def mc_animation():
    new_mc = mc_frames[mc_index]
    new_mc = pygame.transform.scale(new_mc, (130, 150))
    new_mc_rect = new_mc.get_rect(center=(100, mc_rect.centery))
    return new_mc, new_mc_rect


def mc_animation_jump():
    new_mc = mc_jump_frames[mc_jump_index]
    new_mc = pygame.transform.scale(new_mc, (130, 150))
    new_mc_rect = new_mc.get_rect(center=(100, mc_rect.centery))
    return new_mc, new_mc_rect


def mc_animation_death():
    new_mc = mc_death_frames[mc_death_index]
    new_mc = pygame.transform.scale(new_mc, (170, 150))
    new_mc_rect = new_mc.get_rect(center=(100, mc_rect.centery))
    return new_mc, new_mc_rect
```

```
def score_display(game_state):
    if game_state == 'main_game':
        score_surface = game_font.render(f"Score: {(int(score))}", True, (255, 255, 255))
        score_rect = score_surface.get_rect(center=(630, 75))
        screen.blit(score_surface, score_rect)

    if game_state == 'game_over':
        high_score_surface = game_font.render(f"High Score: {(int(high_score))}", True, (255, 255, 255))
        high_score_rect = high_score_surface.get_rect(center=(650, 550))
        screen.blit(high_score_surface, high_score_rect)


def update_score(score, high_score):
    if score > high_score:
        high_score = score
    return high_score


def pipe_score_check():
    global score, can_score

    if pipe_list:
        for pipe in pipe_list:
            if 95 < pipe.centerx < 105 and can_score == True:
                can_score = False
                score += 1
                score_sound.play()
            if pipe.centerx < 0:
                can_score = True
```

```
# General Setup
pygame.init()
pygame.display.set_caption("Steam Man")
screen = pygame.display.set_mode((1280, 720))  # Width, Height
clock = pygame.time.Clock()
game_font = pygame.font.Font("visitor1.ttf", 60)
game_font.set_underline(True)
game_font.set_italic(True)

cover_surface = pygame.image.load("assets/cover.png").convert_alpha()
game_icon = pygame.image.load("assets/mc_run/mc1.png")
pygame.display.set_icon(game_icon)  # Game Icon

bg_surface_swamp = pygame.image.load("assets/backgrounds/bg_swamp.png").convert()
bg_surface_war = pygame.image.load("assets/backgrounds/bg_war.png").convert()
bg_surface_snow = pygame.image.load("assets/backgrounds/bg_snow.png").convert()
bg_surface_desert = pygame.image.load("assets/backgrounds/bg_desert.png").convert()
bg_frames = [bg_surface_swamp, bg_surface_war, bg_surface_snow, bg_surface_desert]
bg_index = 0
bg_surface = bg_frames[bg_index]
bg_surface = pygame.transform.scale(bg_surface, (1280, 720))

BG_CHANGE = pygame.USEREVENT + 4
pygame.time.set_timer(BG_CHANGE, 7500)

floor_surface_swamp = pygame.image.load("assets/floor/floor_swamp.png").convert_alpha()
floor_surface_war = pygame.image.load("assets/floor/floor_war.png").convert_alpha()
floor_surface_snow = pygame.image.load("assets/floor/floor_snow.png").convert_alpha()
floor_surface_desert = pygame.image.load("assets/floor/floor_desert.png").convert_alpha()
floor_frames = [floor_surface_swamp, floor_surface_war, floor_surface_snow, floor_surface_desert]
floor_index = 0
```

```
floor_surface = floor_frames[floor_index]
floor_surface = pygame.transform.scale(floor_surface, (1280, 720))
floor_x_pos = 0

FLOOR_CHANGE = pygame.USEREVENT + 5
pygame.time.set_timer(FLOOR_CHANGE, 7500)

mc_run1 = pygame.transform.scale2x(pygame.image.load('assets/mc_run/mc1.png')).convert_alpha()
mc_run2 = pygame.transform.scale2x(pygame.image.load('assets/mc_run/mc2.png')).convert_alpha()
mc_run3 = pygame.transform.scale2x(pygame.image.load('assets/mc_run/mc3.png')).convert_alpha()
mc_run4 = pygame.transform.scale2x(pygame.image.load('assets/mc_run/mc4.png')).convert_alpha()
mc_run5 = pygame.transform.scale2x(pygame.image.load('assets/mc_run/mc5.png')).convert_alpha()
mc_run6 = pygame.transform.scale2x(pygame.image.load('assets/mc_run/mc6.png')).convert_alpha()
mc_frames = [mc_run1, mc_run2, mc_run3, mc_run4, mc_run5, mc_run6]
mc_index = 0
mc_surface = mc_frames[mc_index]
mc_surface = pygame.transform.scale(mc_surface, (130, 150))
mc_rect = mc_surface.get_rect(center=(100, 560))

MC_RUN = pygame.USEREVENT + 1
pygame.time.set_timer(MC_RUN, 125)

mc_jump1 = pygame.transform.scale2x(pygame.image.load('assets/mc_jump/mc_jump_3.png')).convert_alpha()
mc_jump2 = pygame.transform.scale2x(pygame.image.load('assets/mc_jump/mc_jump_3.png')).convert_alpha()
mc_jump3 = pygame.transform.scale2x(pygame.image.load('assets/mc_jump/mc_jump_4.png')).convert_alpha()
mc_jump4 = pygame.transform.scale2x(pygame.image.load('assets/mc_jump/mc_jump_4.png')).convert_alpha()
mc_jump5 = pygame.transform.scale2x(pygame.image.load('assets/mc_jump/mc_jump_5.png')).convert_alpha()
mc_jump6 = pygame.transform.scale2x(pygame.image.load('assets/mc_jump/mc_jump_5.png')).convert_alpha()
mc_jump_frames = [mc_jump1, mc_jump2, mc_jump3, mc_jump4, mc_jump5, mc_jump6]
mc_jump_index = 0
mc_surface = mc_jump_frames[mc_jump_index]
mc_surface = pygame.transform.scale(mc_surface, (130, 150))
mc_rect = mc_surface.get_rect(center=(100, 560))

MC_JUMP = pygame.USEREVENT + 2
pygame.time.set_timer(MC_JUMP, 175)

mc_death1 = pygame.transform.scale2x(pygame.image.load('assets/mc_death/mc_death_1.png')).convert_alpha()
mc_death2 = pygame.transform.scale2x(pygame.image.load('assets/mc_death/mc_death_2.png')).convert_alpha()
mc_death3 = pygame.transform.scale2x(pygame.image.load('assets/mc_death/mc_death_3.png')).convert_alpha()
mc_death4 = pygame.transform.scale2x(pygame.image.load('assets/mc_death/mc_death_4.png')).convert_alpha()
mc_death5 = pygame.transform.scale2x(pygame.image.load('assets/mc_death/mc_death_5.png')).convert_alpha()
mc_death6 = pygame.transform.scale2x(pygame.image.load('assets/mc_death/mc_death_6.png')).convert_alpha()
mc_death_frames = [mc_death1, mc_death2, mc_death3, mc_death4, mc_death5, mc_death6]
mc_death_index = 0
mc_surface = mc_death_frames[mc_death_index]
mc_surface = pygame.transform.scale(mc_surface, (130, 150))
mc_rect = mc_surface.get_rect(center=(100, 560))
MC_DEATH = pygame.USEREVENT + 3
pygame.time.set_timer(MC_DEATH, 175)

pipe_surface = pygame.transform.scale2x(pygame.image.load('assets/pipe.png'))
pipe_list = []
pipe_height = [275, 300, 425, 600, 850]
SPAWNPIPE = pygame.USEREVENT
pygame.time.set_timer(SPAWNPIPE, 1750)

bird_surface = pygame.transform.scale2x(pygame.image.load('assets/bird/bird_1.png')).convert_alpha()
bird_surface = pygame.transform.scale(bird_surface, (50, 50))
bird_list = []
bird_altitude = [0, 250, 250, 550]
```

```
SPAWNBIRD = pygame.USEREVENT + 6
pygame.time.set_timer(SPAWNBIRD, 3000)

cover_surface = pygame.image.load("assets/cover.png").convert_alpha()
cover_rect = cover_surface.get_rect(center=(640, 360))

jump_sound = pygame.mixer.Sound("assets/sfx/jump2.wav")
death_sound = pygame.mixer.Sound("assets/sfx/death_sound.mp3")
death_sound.set_volume(0.10)
score_sound = pygame.mixer.Sound("assets/sfx/score_sound2.mp3")
score_sound.set_volume(0.10)

pygame.mixer.music.load("assets/sfx/bg_music_swamp.mp3")
pygame.mixer.music.set_volume(0.1)

# Game Variables
gravity = 0.25
mc_movement = 0
jump_or_not = False
can_jump = 3
death = False
game_active = False
can_score = True
score = 0
high_score = 0
speed_factor = 1

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

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE and game_active == True and can_jump > 0:  # The player has 3 jumps
                jump_or_not = True
                can_jump -= 1
                mc_movement = 0
                mc_movement -= 9
                jump_sound.play()

            if event.key == pygame.K_SPACE and game_active == False:  # Game Reset
                bg_surface = pygame.transform.scale(bg_surface_swamp, (1280, 720))
                floor_surface = pygame.transform.scale(floor_surface_swamp, (1280, 720))
                pygame.mixer.music.load("assets/sfx/bg_music_swamp.mp3")
                pygame.mixer.music.play(-1)
                game_active = True
                death = False
                mc_death_index = 0
                mc_rect.center = (100, 560)
                bird_list.clear()
                pipe_list.clear()
                score = 0
                speed_factor = 1

        if event.type == BG_CHANGE and game_active == True:
            if bg_index < 3:
                bg_index += 1
            else:
                bg_index = 0

            bg_surface = bg_and_music_change()
```

```
        if event.type == FLOOR_CHANGE and game_active == True:
            if floor_index < 3:
                floor_index += 1
            else:
                floor_index = 0

            floor_surface = floor_change()

        if event.type == SPAWNPIPE:
            pipe_list.extend(create_pipe())

        if event.type == SPAWNBIRD:
            bird_list.append(create_bird())

        if death == True:
            bg_index = 0
            floor_index = 0
            if event.type == MC_DEATH:
                if mc_death_index < 5:
                    mc_death_index += 1
                mc_surface, mc_rect = mc_animation_death()


        elif jump_or_not == True and death == False:
            if event.type == MC_JUMP:
                if mc_jump_index < 5:
                    mc_jump_index += 1
                else:
                    mc_jump_index = 0

                mc_surface, mc_rect = mc_animation_jump()
        else:
            if event.type == MC_RUN:
                if mc_index < 5:
                    mc_index += 1
                else:
                    mc_index = 0

                mc_surface, mc_rect = mc_animation()

    screen.blit(bg_surface, (0, 0))

    if game_active:

        if check_collision_bird(bird_list) == False or check_collision(pipe_list) == False:
            game_active = False
            pygame.mixer.music.stop()

        # Main Character
        screen.blit(mc_surface, mc_rect)
        mc_movement += gravity
        mc_rect.centery += mc_movement
        if mc_rect.centery >= 560:
            mc_rect.centery = 560
            can_jump = 3
            jump_or_not = False
        elif mc_rect.centery <= 20:
            mc_rect.centery = 20

        # Pipes
        pipe_list = move_pipes(pipe_list)
        draw_pipes(pipe_list)
```

```
        check_collision_bird(bird_list)

        # Score
        pipe_score_check()
        score_display('main_game')

        # Floor Speed
        floor_x_pos -= 4 * speed_factor

        # Speed Factor
        speed_factor += 0.00015

    else:
        mc_movement += gravity
        mc_rect.centery += mc_movement
        if mc_rect.centery >= 555:
            mc_rect.centery = 555
        screen.blit(mc_surface, mc_rect)
        screen.blit(cover_surface, cover_rect)
        high_score = update_score(score, high_score)
        score_display('game_over')

        # Floor Speed
        floor_x_pos -= 1

    # Floor
    draw_floor()
    if floor_x_pos <= -1280:
        floor_x_pos = 0

    pygame.display.update()
    clock.tick(120)
```