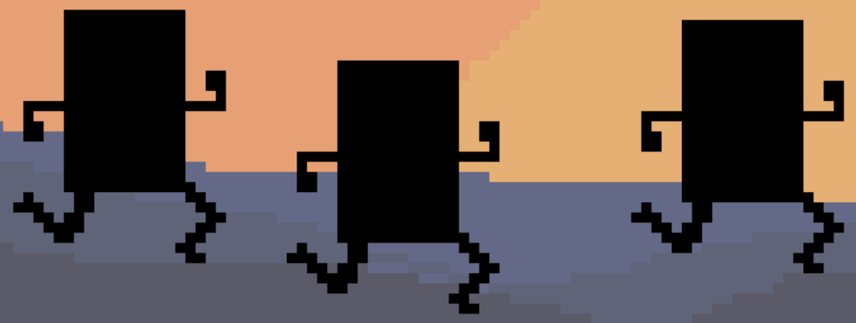
roadrunner

J*B TAKEOVER

MARYAM, JUSTIN, MICAIAH, ELAINE



THE CONCEPT

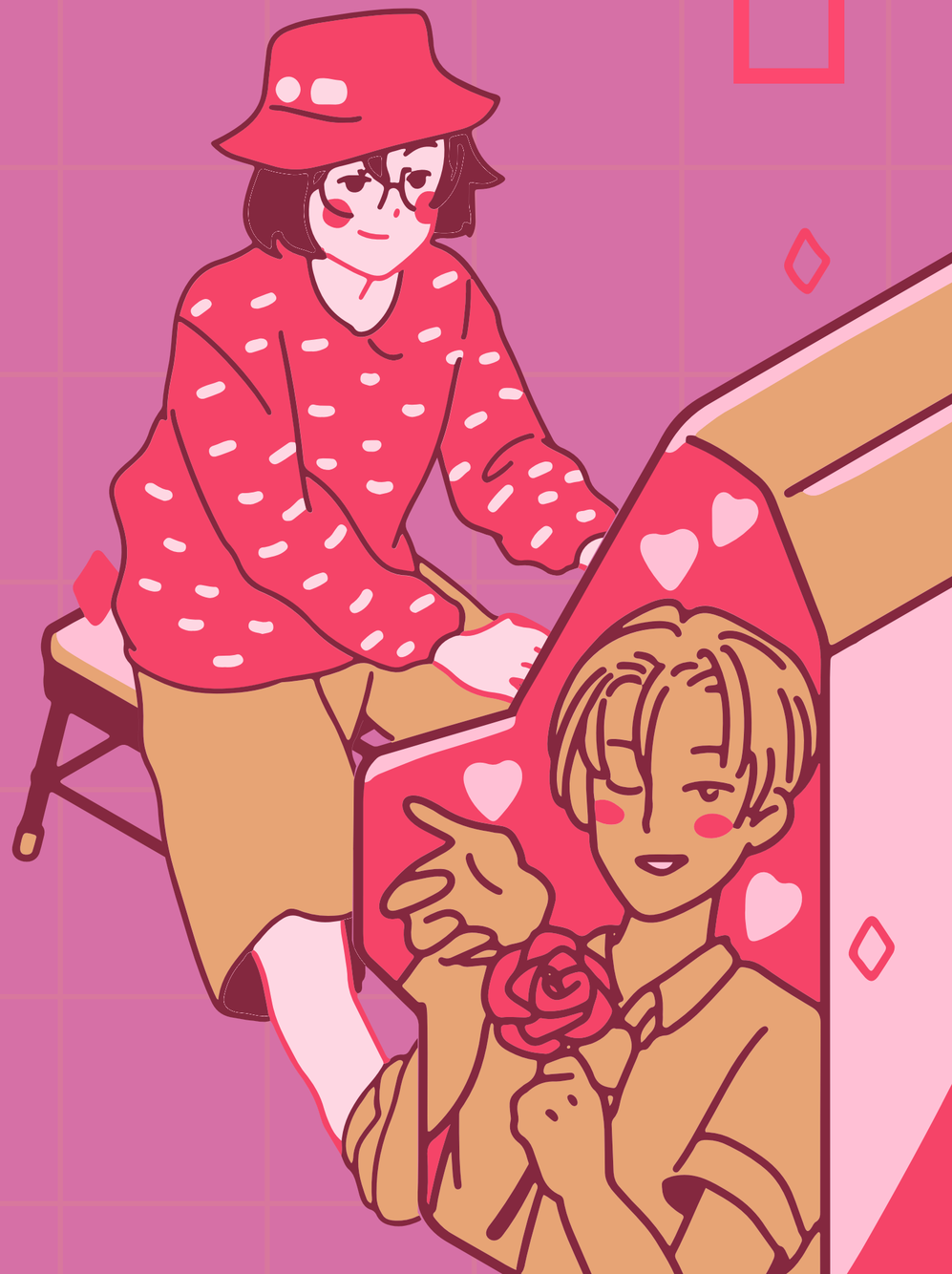
A COMPUTER SCIENCE GRADUATE FROM THE UNIVERSITY OF DELAWARE FINDS HIMSELF IN THE JOB MARKET. HE NEEDS HELP TO AVOID EVIL JOB APPLICATIONS AND BEAT THE FINAL JOB APPLICATION BOSS. THE PLAYER HAS TO NAVIGATE THROUGH LEVELS AND COLLECT AS MANY COINS AS POSSIBLE TO HAVE A HIGHER CHANCE AT BEATING THE BOSS.

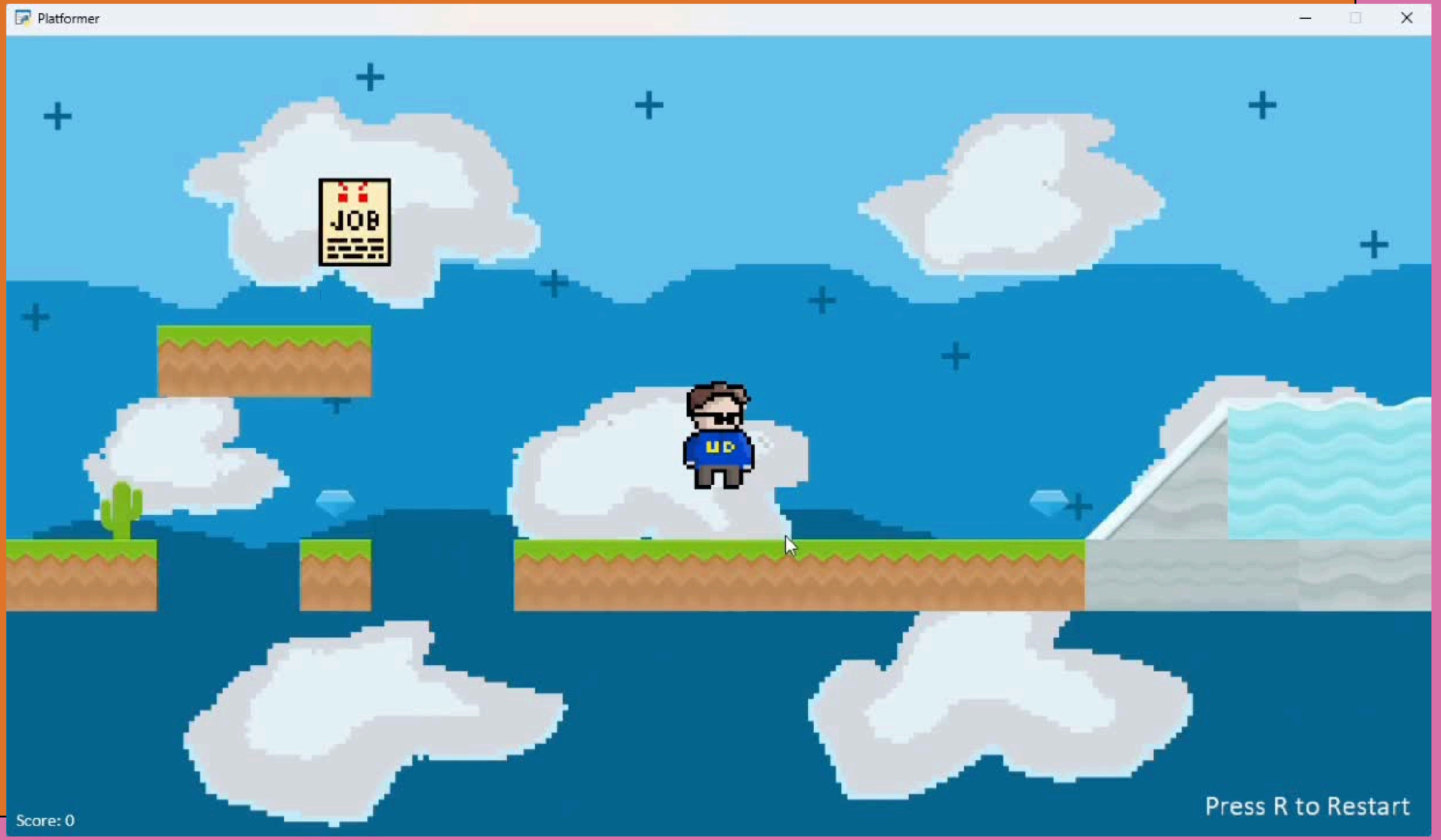
CHALLENGES DURING THE PROCESS

- FIRST STEPS
- TIME RESTRICTIONS
- COIN SPEED UPDATES
- UPLOADING CUSTOM FILES

GAME DEMO

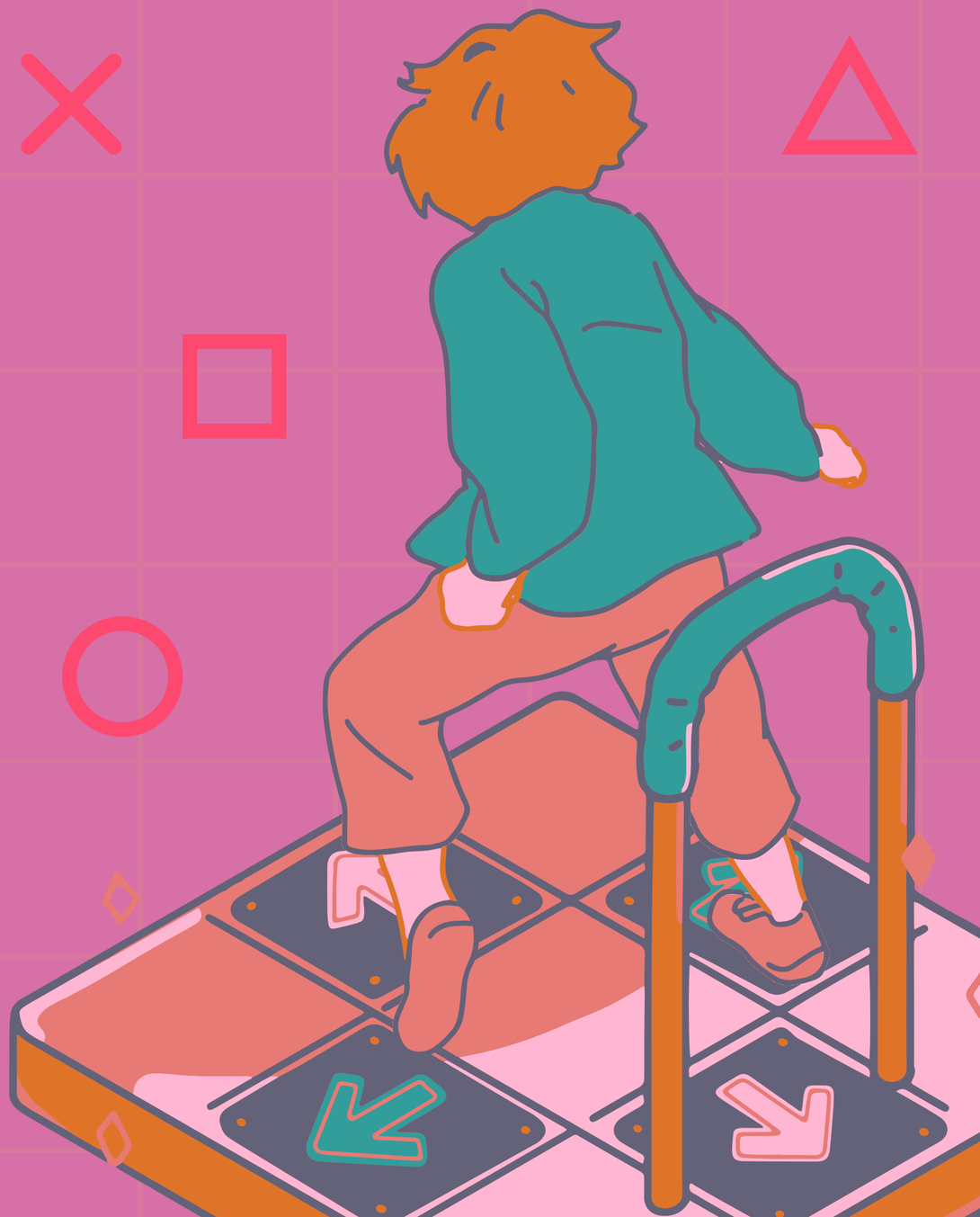
WATCH A QUICK PLAYTHROUGH!




Platformer
JOB
Score: 0
Press R to Restart

THE CODE

- CLASSES
- FUNCTIONS
- FOR LOOPS
- CHALLENGES
- ADVANCED FEATURES



```python
class StartView(arcade.View):

    def on_show_view(self):
        self.sprites = arcade.SpriteList()

        # Load the fullscreen start image as a sprite
        sprite = arcade.Sprite("titlescreen.png")

        # Force it to fill the window
        sprite.width = WINDOW_WIDTH
        sprite.height = WINDOW_HEIGHT

        # Center the sprite
        sprite.center_x = WINDOW_WIDTH // 2
        sprite.center_y = WINDOW_HEIGHT // 2

        # Add it to the SpriteList to draw it
        self.sprites.append(sprite)

    def on_draw(self):
        self.clear()
        self.sprites.draw()

    def on_key_press(self, key, modifiers):
        if key == arcade.key.SPACE:
            game = GameView(level=1)
            game.setup()
            self.window.show_view(game)
```

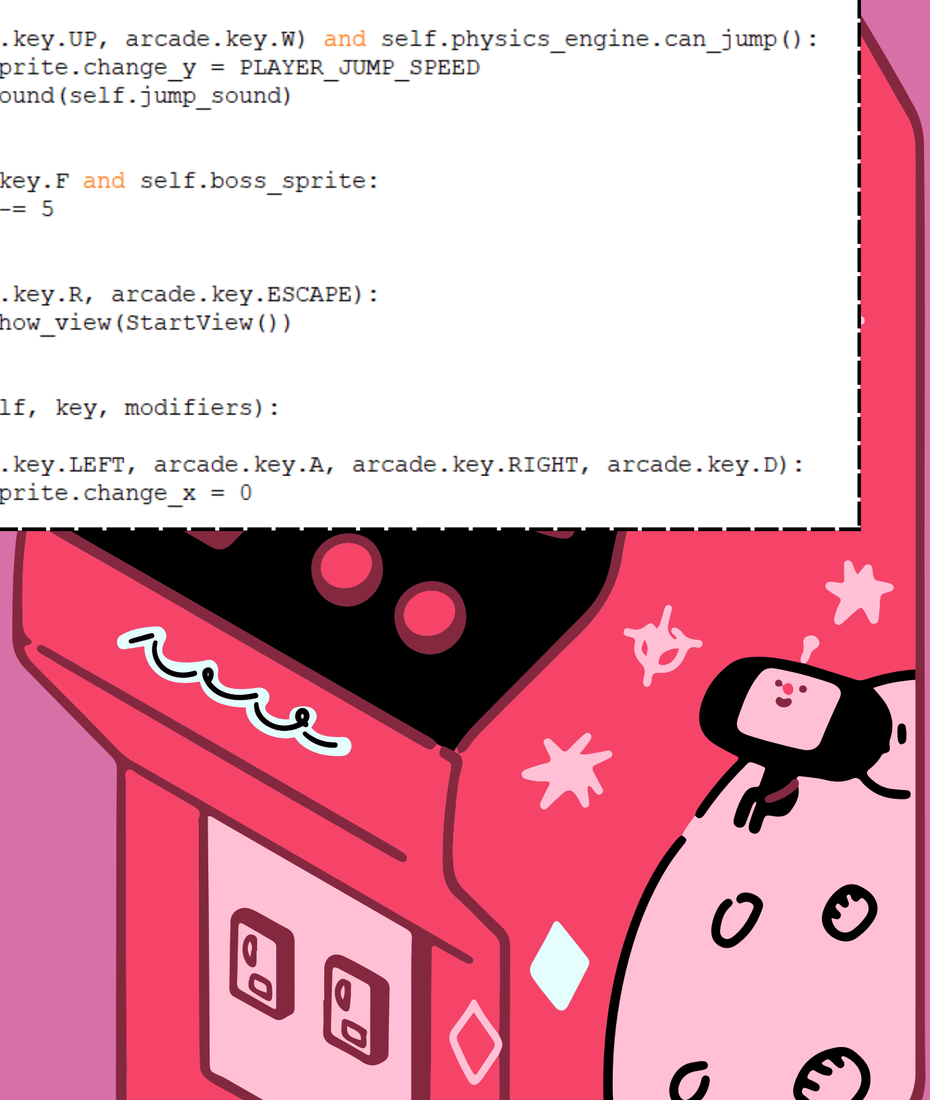
```python
class WinView(arcade.View):

    def __init__(self, final_score):
        super().__init__()
        self.final_score = final_score
        self.sprites = arcade.SpriteList()

    def on_show_view(self):
        win_sprite = arcade.Sprite("winscreen.png")
        win_sprite.width = WINDOW_WIDTH
        win_sprite.height = WINDOW_HEIGHT
        win_sprite.center_x = WINDOW_WIDTH // 2
        win_sprite.center_y = WINDOW_HEIGHT // 2
        self.sprites.append(win_sprite)

        # Score text (left side, centered vertically)
        self.score_text = arcade.Text(
            f"{self.final_score}",
            x=700,
            y=400,
            color=arcade.color.WHITE,
            font_size=50,
            anchor_x="left",
            anchor_y="center"
        )

    def on_draw(self):
        self.clear()
        self.sprites.draw()
        self.score_text.draw()

    def on_key_press(self, key, modifiers):
        if key == arcade.key.SPACE:
            start = StartView()
            self.window.show_view(start)
```

```python
class GameView(arcade.View):

    def __init__(self, level):
        super().__init__()

        self.level = level

        self.player_sprite = None

        # Fixed speed (no more boosting)
        self.player_speed = PLAYER_MOVEMENT_SPEED

        # MAP / SCENE
        self.tile_map = None
        self.scene = None

        # CAMERAS
        self.camera = None
        self.gui_camera = None

        # SCORE
        self.score = 0
        self.score_text = None
        self.end_of_map = 0

        # Level 3 survival
        self.level3_timer = 0

        # BACKGROUND
        self.background_list = arcade.SpriteList()

        # ENEMY
        self.enemy_sprite = None
        self.enemy_started = False
        self.enemy_speed = 7

        # BOSS
        self.boss_sprite = None
        self.boss_hp = 100
        self.flash_timer = 0
        self.show_flash = True

        # SOUNDS
        self.collect_coin_sound = arcade.load_sound(":resources:sounds/coin1.wav")
        self.jump_sound = arcade.load_sound(":resources:sounds/jump1.wav")
        self.gameover_sound = arcade.load_sound(":resources:sounds/gameover1.wav")
```

```python
    def setup(self):
        # LOAD MAP
        layer_options = {"Platforms": {"use_spatial_hash": True}}
        self.tile_map = arcade.load_tilemap(
            f"map2_level_{self.level}.json",
            scaling=TILE_SCALING,
            layer_options=layer_options,
        )
        self.scene = arcade.Scene.from_tilemap(self.tile_map)
        # BACKGROUND
        bg = arcade.Sprite("mainbackground.png")
        bg.width = WINDOW_WIDTH
        bg.height = WINDOW_HEIGHT
        bg.center_x = WINDOW_WIDTH // 2
        bg.center_y = WINDOW_HEIGHT // 2
        self.background_list.append(bg)
        # PLAYER
        self.player_sprite = arcade.Sprite("my_player.png", scale=0.09)
        self.player_sprite.center_x = 128
        self.player_sprite.center_y = 128
        self.scene.add_sprite("Player", self.player_sprite)
        # ENEMY (ONLY LEVELS 1-2)
        if self.level < 3:
            self.enemy_sprite = arcade.Sprite("regularjobapp.png", scale=0.09)
            self.enemy_sprite.center_x = self.player_sprite.center_x - 200
            self.enemy_sprite.center_y = self.player_sprite.center_y
            self.scene.add_sprite("Enemy", self.enemy_sprite)
        else:
            self.enemy_sprite = None
        # BOSS (LEVEL 3 ONLY)
        if self.level == 3:
            self.boss_sprite = arcade.Sprite("bossjob.png", scale=0.15)
            self.boss_sprite.center_x = (
                self.tile_map.width * self.tile_map.tile_width * TILE_SCALING - 150
            )
            self.boss_sprite.center_y = 200
            self.scene.add_sprite("Boss", self.boss_sprite)
            self.boss_hp = 100
        # PHYSICS
        self.physics_engine = arcade.PhysicsEnginePlatformer(
            self.player_sprite,
            walls=self.scene["Platforms"],
            gravity_constant=GRAVITY
        )
        # CAMERAS
        self.camera = arcade.Camera2D()
        self.gui_camera = arcade.Camera2D()
        # SCORE TEXT
        self.score_text = arcade.Text(f"Score: {self.score}", x=10, y=10)
```

```python
def on_draw(self):
    self.clear()
    # Background
    self.background_list.draw()
    # Game world
    self.camera.use()
    self.scene.draw()
    # GUI
    self.gui_camera.use()
    self.score_text.draw()
    # Restart text
    arcade.draw_text(
        "Press R to Restart",
        WINDOW_WIDTH - 20,
        20,
        arcade.color.WHITE,
        20,
        anchor_x="right"
    )
    # BOSS UI
    if self.boss_sprite:
        # HP bar
        bar_width = 400
        bar_height = 20
        x = WINDOW_WIDTH // 2
        y = WINDOW_HEIGHT - 40

        left = x - bar_width // 2
        right = x + bar_width // 2
        top = y + bar_height // 2
        bottom = y - bar_height // 2

        arcade.draw_lrbt_rectangle_filled(left, right, bottom, top, arcade.color.DARK_GRAY)

        hp_width = (self.boss_hp / 100) * bar_width
        arcade.draw_lrbt_rectangle_filled(left, left + hp_width, bottom, top, arcade.color.RED)

        # Flashing attack text
        if self.show_flash:
            arcade.draw_text(
                "PRESS F TO ATTACK",
                WINDOW_WIDTH // 2,
                WINDOW_HEIGHT // 2 + 200,
                arcade.color.WHITE,
                30,
                anchor_x="center"
            )
```

```python
def on_key_press(self, key, modifiers):

    # Movement
    if key in (arcade.key.LEFT, arcade.key.A):
        self.enemy_started = True
        self.player_sprite.change_x = -self.player_speed

    if key in (arcade.key.RIGHT, arcade.key.D):
        self.enemy_started = True
        self.player_sprite.change_x = self.player_speed

    if key in (arcade.key.UP, arcade.key.W) and self.physics_engine.can_jump():
        self.player_sprite.change_y = PLAYER_JUMP_SPEED
        arcade.play_sound(self.jump_sound)

    # Attack boss
    if key == arcade.key.F and self.boss_sprite:
        self.boss_hp -= 5

    # Reset
    if key in (arcade.key.R, arcade.key.ESCAPE):
        self.window.show_view(StartView())


def on_key_release(self, key, modifiers):

    if key in (arcade.key.LEFT, arcade.key.A, arcade.key.RIGHT, arcade.key.D):
        self.player_sprite.change_x = 0
```



```python
# BOSS MOVEMENT
if self.boss_sprite:
    if self.boss_sprite.center_x < self.player_sprite.center_x:
        self.boss_sprite.center_x += 6
    else:
        self.boss_sprite.center_x -= 6

    if self.boss_sprite.center_y < self.player_sprite.center_y:
        self.boss_sprite.center_y += 3
    else:
        self.boss_sprite.center_y -= 3
# Boss kills player on contact
if self.boss_sprite and arcade.check_for_collision(self.player_sprite, self.boss_sprite):
    arcade.play_sound(self.gameover_sound)
    self.setup()
# BOSS DEFEATED
if self.boss_sprite and self.boss_hp <= 0:
    self.window.show_view(WinView(self.score))
    return

# COIN COLLECTION
for coin in arcade.check_for_collision_with_list(self.player_sprite, self.scene["Coins"]):
    coin.remove_from_sprite_lists()
    self.score += 75
    self.score_text.text = f"Score: {self.score}"
    arcade.play_sound(self.collect_coin_sound)
# DEATH FROM HAZARD
if arcade.check_for_collision_with_list(self.player_sprite, self.scene["Don't Touch"]):
    arcade.play_sound(self.gameover_sound)
    self.setup()
# LEVEL COMPLETE
if self.player_sprite.center_x >= self.end_of_map:
    next_level = GameView(self.level + 1)
    next_level.setup()
    self.window.show_view(next_level)
# Camera follows player
self.camera.position = self.player_sprite.position
```

```python
# ENEMY CHASE
if self.enemy_sprite and self.enemy_started:
    if self.enemy_sprite.center_x < self.player_sprite.center_x:
        self.enemy_sprite.center_x += self.enemy_speed
    else:
        self.enemy_sprite.center_x -= self.enemy_speed

    if self.enemy_sprite.center_y < self.player_sprite.center_y:
        self.enemy_sprite.center_y += self.enemy_speed * 0.5
    else:
        self.enemy_sprite.center_y -= self.enemy_speed * 0.5
# Enemy collision kills player
if self.enemy_sprite and arcade.check_for_collision(self.player_sprite, self.enemy_sprite):
    arcade.play_sound(self.gameover_sound)
    self.setup()
```

THANK YOU

ENJOY YOUR GAME!