DESCRIPTION

PROBLEM DEFINITION:
Dark Ark is software where users get to the middle in up,down,left and right by not hit the black things in the circular path

OBJECTIVE:
The aim of the software is to give complete fun ,thinking where to move the white thing at the right time by avoid the black thing

METHODOLOGY:

TITLE CARD:
The user click the start icon at bottom of the screen,

GAME PAGE:
The game startes with a image then two circles appears and a gap is create between them.Move the white thing up,dowm,left or right add score to it by avoid the black thing in the circular path

END GAME:
When the white thing hit the black thing then a game over image will pop up and the user can click the retry again icon at bottom of the screen

TOOLS/PLATFORMS USED:
Python Programming language is used as the back end and Pygame is used as the front end for implementing the project

SOURCE CODE

```python
import pygame
import random
import math

SCREEN = WIDTH, HEIGHT = 288, 512
CENTER = WIDTH //2, HEIGHT // 2
MAX_RAD = 120
```

```python
pygame.font.init()
pygame.mixer.init()
```

```python
class Player:
    def __init__(self, win):
        self.win = win
        self.reset()

    def update(self, player_alive, color, shadow_group):
        if player_alive:
            if self.x <= CENTER[0] - MAX_RAD or self.x >= CENTER[0] + MAX_RAD or \
                self.y <= CENTER[1] - MAX_RAD or self.y >= CENTER[1] + MAX_RAD:
                    if self.dx:
                        self.dx *= -1
                    elif self.dy:
                        self.dy *= -1
```

```python
                    shadow_group.empty()
```

```python
            if self.index == 1 and self.y > CENTER[1]:
                    self.reset_pos()
                    self.can_move = True
            elif self.index == 2 and self.x < CENTER[0]:
                    self.reset_pos()
                    self.can_move = True
            elif self.index == 3 and self.y < CENTER[1]:
                    self.reset_pos()
                    self.can_move = True
            elif self.index == 4 and self.x > CENTER[0]:
                    self.reset_pos()
                    self.can_move = True
```

```python
            self.x += self.dx
            self.y += self.dy
```

```python
            self.rect = pygame.draw.circle(self.win, (255, 255, 255), (self.x, self.y), 6)
            pygame.draw.circle(self.win, color, (self.x, self.y), 3)
```

```python
    def set_move(self, index):
        if self.can_move:
            self.index = index
            if index == 1:
                self.dy = -self.vel
            if index == 2:
                self.dx = self.vel
            if index == 3:
                self.dy = self.vel
            if index == 4:
                self.dx = -self.vel

            self.can_move = False

    def reset_pos(self):
        self.x = CENTER[0]
        self.y = CENTER[1]
        self.dx = self.dy = 0

    def reset(self):
        self.x = CENTER[0]
        self.y = CENTER[1]
        self.vel = 6

        self.index = None
        self.dx = self.dy = 0
        self.can_move = True

class Dot(pygame.sprite.Sprite):
    def __init__(self, x, y, win):
        super(Dot, self).__init__()

        self.x = x
        self.y = y
        self.color = (255, 255, 255)
        self.win = win

        self.rect = pygame.draw.circle(win, self.color, (x,y), 6)

    def update(self):
        pygame.draw.circle(self.win, self.color, (self.x,self.y), 6)
        self.rect = pygame.draw.circle(self.win, self.color, (self.x,self.y), 6)

class ShadowImage:
    def __init__(self):
        self.image = pygame.Surface((10, 100), pygame.SRCALPHA)
        self.image.fill((255, 255, 255, 100))
        self.rect = self.image.get_rect()

    def rotate(self, angle):
        rotated = pygame.transform.rotate(self.image, angle)
        self.rect = rotated.get_rect()
        return rotated


class Shadow(pygame.sprite.Sprite):
    def __init__(self, index, win):
        super(Shadow, self).__init__()

        self.index = index
        self.win = win
        self.color = (255, 255, 255)
        self.shadow = ShadowImage()


        if self.index == 1:
            self.image = self.shadow.rotate(0)
            self.x = CENTER[0] - 5
            self.y = CENTER[1] - MAX_RAD + 10
        if self.index == 2:
            self.image = self.shadow.rotate(90)
            self.x = CENTER[0] + 10
            self.y = CENTER[1] - 5
        if self.index == 3:
            self.image = self.shadow.rotate(0)
            self.x = CENTER[0] - 5
            self.y = CENTER[1] + 10
        if self.index == 4:
            self.image = self.shadow.rotate(-90)
            self.x = CENTER[0] - MAX_RAD + 10
            self.y = CENTER[1] - 5

    def update(self):
        self.win.blit(self.image, (self.x,self.y))
```

```
class Balls(pygame.sprite.Sprite):
    def __init__(self, pos, type_, inverter, win):
        super(Balls, self).__init__()

        self.initial_pos = pos
        self.color = (0,0,0)
        self.type = type_
        self.inverter = inverter
        self.win = win
        self.reset()
```

```
        self.rect = pygame.draw.circle(self.win, self.color, (self.x,self.y), 6)
```

```
    def update(self):
        dx = 0
        x = round(CENTER[0] + self.radius * math.cos(self.angle * math.pi / 180))
        y = round(CENTER[1] + self.radius * math.sin(self.angle * math.pi / 180))
```

```
        self.angle += self.dtheta
        if self.dtheta == 1 and self.angle >= 360:
            self.angle = 0
        elif self.dtheta == -1 and self.angle <= 0:
            self.angle = 360
```

```
        self.rect = pygame.draw.circle(self.win, self.color, (x,y), 6)
```

```
    def reset(self):
        self.x, self.y = self.initial_pos
        if self.type == 1:
```

```
            if self.x == CENTER[0]-105:
                self.angle = 180
            if self.x == CENTER[0]+105:
                self.angle = 0
            if self.x == CENTER[0]-45:
                self.angle = 180
            if self.x == CENTER[0]+45:
                self.angle = 0
```

```
            self.radius = abs(CENTER[0] - self.x) - 3
            self.dtheta = 1
```

```
        elif self.type == 2:
            if self.y == CENTER[1] - 75:
                self.angle = 90
            if self.y == CENTER[1] + 75:
                self.angle = 270
```

```
            self.radius = abs(CENTER[1] - self.y) - 3
            self.dtheta = -1
```

```
class Particle(pygame.sprite.Sprite):
    def __init__(self, x, y, color, win):
        super(Particle, self).__init__()
        self.x = x
        self.y = y
        self.color = color
        self.win = win
        self.size = random.randint(4,7)
        xr = (-3,3)
        yr = (-3,3)
        f = 2
        self.life = 40
        self.x_vel = random.randrange(xr[0], xr[1]) * f
        self.y_vel = random.randrange(yr[0], yr[1]) * f
        self.lifetime = 0

    def update (self):
        self.size -= 0.1
        self.lifetime += 1
        if self.lifetime <= self.life:
            self.x += self.x_vel
            self.y += self.y_vel
            s = int(self.size)
            pygame.draw.rect(self.win, self.color, (self.x, self.y,s,s))
        else:
            self.kill()
```

```
class Message:
    def __init__(self, x, y, size, text, font, color, win):
        self.win = win
        self.color = color
        self.x, self.y = x, y
        if not font:
            self.font = pygame.font.SysFont("Verdana", size)
            anti_alias = True
        else:
            self.font = pygame.font.Font(font, size)
            anti_alias = False
        self.image = self.font.render(text, anti_alias, color)
        self.rect = self.image.get_rect(center=(x,y))
        self.shadow = self.font.render(text, anti_alias, (54,69,79))
        self.shadow_rect = self.image.get_rect(center=(x+2,y+2))

    def update(self, text=None, shadow=True):
        if text:
            self.image = self.font.render(f"{text}", False, self.color)
            self.rect = self.image.get_rect(center=(self.x,self.y))
            self.shadow = self.font.render(f"{text}", False, (54,69,79))
            self.shadow_rect = self.image.get_rect(center=(self.x+2,self.y+2))
        if shadow:
```

```python
            self.win.blit(self.shadow, self.shadow_rect)
        self.win.blit(self.image, self.rect)

class BlinkingText(Message):
    def __init__(self, x, y, size, text, font, color, win):
        super(BlinkingText, self).__init__(x, y, size, text, font, color, win)
        self.index = 0
        self.show = True

    def update(self):
        self.index += 1
        if self.index % 40 == 0:
            self.show = not self.show

        if self.show:
            self.win.blit(self.image, self.rect)

class Button(pygame.sprite.Sprite):
    def __init__(self, img, scale, x, y):
        super(Button, self).__init__()

        self.scale = scale
        self.image = pygame.transform.scale(img, self.scale)
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

        self.clicked = False

    def update_image(self, img):
        self.image = pygame.transform.scale(img, self.scale)

    def draw(self, win):
        action = False
        pos = pygame.mouse.get_pos()
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] and not self.clicked:
                action = True
                self.clicked = True

            if not pygame.mouse.get_pressed()[0]:
                self.clicked = False

        win.blit(self.image, self.rect)
        return action
```

```python
# Arc Dash
import random
import pygame

from objects import Player, Balls, Dot, Shadow, Particle, Message, BlinkingText, Button

pygame.init()
SCREEN = WIDTH, HEIGHT = 288, 512
CENTER = WIDTH //2, HEIGHT // 2

info = pygame.display.Info()
```

```python
width = info.current_w
height = info.current_h
```

```python
if width >= height:
    win = pygame.display.set_mode(SCREEN, pygame.NOFRAME)
else:
    win = pygame.display.set_mode(SCREEN, pygame.NOFRAME | pygame.SCALED | pygame.FULLSCREEN)
```

```python
clock = pygame.time.Clock()
FPS = 60
```

```python
# COLORS **********************************************************************
```

```python
RED = (255,0,0)
GREEN = (0,177,64)
BLUE = (30, 144,255)
ORANGE = (252,76,2)
YELLOW = (254,221,0)
PURPLE = (155,38,182)
AQUA = (0,103,127)
WHITE = (255,255,255)
BLACK = (0,0,0)
```

```python
color_list = [RED, GREEN, BLUE, ORANGE, YELLOW, PURPLE]
color_index = 0
color = color_list[color_index]
```

```python
# FONTS ***********************************************************************
```

```python
title_font = "Fonts/Aladin-Regular.ttf"
tap_to_play_font = "Fonts/BubblegumSans-Regular.ttf"
score_font = "Fonts/DalelandsUncialBold-82zA.ttf"
game_over_font = "Fonts/ghostclan.ttf"
```

```python
# MESSAGES ********************************************************************
```

```python
arc = Message(WIDTH-90, 200, 80, "Arc", title_font, WHITE, win)
dash = Message(80, 300, 60, "Dash", title_font, WHITE, win)
tap_to_play = BlinkingText(WIDTH//2, HEIGHT-60, 20, "Tap To Play", tap_to_play_font, WHITE,
win)
game_msg = Message(80, 150, 40, "GAME", game_over_font, BLACK, win)
over_msg = Message(210, 150, 40, "OVER!", game_over_font, WHITE, win)
score_text = Message(90, 230, 20, "SCORE", None, BLACK, win)
best_text = Message(200, 230, 20, "BEST", None, BLACK, win)
```

```python
score_msg = Message(WIDTH-60, 50, 50, "0", score_font, WHITE, win)
final_score_msg = Message(90, 280, 40, "0", tap_to_play_font, BLACK, win)
high_score_msg = Message(200, 280, 40, "0", tap_to_play_font, BLACK, win)
```

```python
# SOUNDS **********************************************************************
```

```python
score_fx = pygame.mixer.Sound('Sounds/point.mp3')
death_fx = pygame.mixer.Sound('Sounds/dead.mp3')
score_page_fx = pygame.mixer.Sound('Sounds/score_page.mp3')
```

```python
pygame.mixer.music.load('Sounds/hk.mp3')
pygame.mixer.music.play(loops=-1)
pygame.mixer.music.set_volume(0.5)
```

```python
# Button images
```

```
home_img = pygame.image.load('Assets/homeBtn.png')
replay_img = pygame.image.load('Assets/replay.png')
sound_off_img = pygame.image.load("Assets/soundOffBtn.png")
sound_on_img = pygame.image.load("Assets/soundOnBtn.png")
```

```
# Buttons
```

```
home_btn = Button(home_img, (24, 24), WIDTH // 4 - 18, 390)
replay_btn = Button(replay_img, (36,36), WIDTH // 2  - 18, 382)
sound_btn = Button(sound_on_img, (24, 24), WIDTH - WIDTH // 4 - 18, 390)
```

```
# GAME VARIABLES **************************************************************
```

```
MAX_RAD = 120
rad_delta = 50
```

```
# OBJECTS *********************************************************************
```

```
ball_group = pygame.sprite.Group()
dot_group = pygame.sprite.Group()
shadow_group = pygame.sprite.Group()
particle_group = pygame.sprite.Group()
p = Player(win)
```

```
ball_positions = [(CENTER[0]-105, CENTER[1]), (CENTER[0]+105, CENTER[1]),
                    (CENTER[0]-45, CENTER[1]), (CENTER[0]+45, CENTER[1]),
                    (CENTER[0], CENTER[1]-75), (CENTER[0], CENTER[1]+75)]
for index, pos in enumerate(ball_positions):
    if index in (0,1):
        type_ = 1
        inverter = 5
    if index in (2,3):
        type_ = 1
        inverter = 3
    if index in (4,5):
        type_ = 2
        inverter = 1
    ball = Balls(pos, type_, inverter, win)
    ball_group.add(ball)
```

```
dot_list = [(CENTER[0], CENTER[1]-MAX_RAD+3), (CENTER[0]+MAX_RAD-3, CENTER[1]),
            (CENTER[0], CENTER[1]+MAX_RAD-3), (CENTER[0]-MAX_RAD+3, CENTER[1])]
dot_index = random.choice([1,2,3,4])
dot_pos = dot_list[dot_index-1]
dot = Dot(*dot_pos, win)
dot_group.add(dot)
```

```
shadow = Shadow(dot_index, win)
shadow_group.add(shadow)
```

```
# VARIABLES *******************************************************************
```

```
clicked = False
num_clicks = 0
player_alive = True
sound_on = True
```

```
score = 0
```

```python
highscore = 0
```

```python
home_page = True
game_page = False
score_page = False
```

```python
running = True
while running:
    win.fill(color)
```

```python
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

```python
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE or \
                event.key == pygame.K_q:
                running = False
```

```python
        if event.type == pygame.MOUSEBUTTONDOWN and home_page:
            home_page = False
            game_page = True
            score_page = False
```

```python
            rad_delta = 50
            clicked = True
            score = 0
            num_clicks = 0
            player_alive = True
```

```python
        if event.type == pygame.MOUSEBUTTONDOWN and game_page:
            if not clicked:
                clicked = True
                for ball in ball_group:
                    if num_clicks % ball.inverter == 0:
                        ball.dtheta *= -1
```

```python
                p.set_move(dot_index)
```

```python
                num_clicks += 1
                if num_clicks % 5 == 0:
                    color_index += 1
                    if color_index > len(color_list) - 1:
                        color_index = 0
```

```python
                    color = color_list[color_index]
```

```python
        if event.type == pygame.MOUSEBUTTONDOWN and game_page:
            clicked = False
```

```python
    if home_page:
        for radius in [30, 60, 90, 120]:
            pygame.draw.circle(win, (0,0,0), CENTER, radius, 8)
            pygame.draw.circle(win, (255,255,255), CENTER, radius, 5)
```

```python
        pygame.draw.rect(win, color, [CENTER[0]-10, CENTER[1]-MAX_RAD, MAX_RAD+50, MAX_RAD])
        pygame.draw.rect(win, color, [CENTER[0]-MAX_RAD, CENTER[1]-10, MAX_RAD, MAX_RAD+50])
```

```python
        arc.update()
        dash.update()
        tap_to_play.update()
```

```
    if score_page:
        game_msg.update()
        over_msg.update()
        score_text.update(shadow=False)
        best_text.update(shadow=False)
```

```
        final_score_msg.update(score, shadow=False)
        high_score_msg.update(highscore, shadow=False)
```

```
        if home_btn.draw(win):
            home_page = True
            score_page = False
            game_page = False
            score = 0
            score_msg = Message(WIDTH-60, 50, 50, "0", score_font, WHITE, win)
```

```
        if replay_btn.draw(win):
            home_page = False
            score_page = False
            game_page = True
```

```
            player_alive = True
            score = 0
            score_msg = Message(WIDTH-60, 50, 50, "0", score_font, WHITE, win)
            p = Player(win)
```

```
        if sound_btn.draw(win):
            sound_on = not sound_on

            if sound_on:
                sound_btn.update_image(sound_on_img)
                pygame.mixer.music.play(loops=-1)
            else:
                sound_btn.update_image(sound_off_img)
                pygame.mixer.music.stop()
```

```
    if game_page:
```

```
        for radius in [30 + rad_delta, 60 + rad_delta, 90 + rad_delta, 120 + rad_delta]:
            if rad_delta > 0:
                radius -= 1
                rad_delta -= 1
            pygame.draw.circle(win, (0,0,0), CENTER, radius, 5)
```

```
        pygame.draw.rect(win, color, [CENTER[0]-10, CENTER[1]-MAX_RAD, 20, MAX_RAD*2])
        pygame.draw.rect(win, color, [CENTER[0]-MAX_RAD, CENTER[1]-10, MAX_RAD*2, 20])
```

```
        if rad_delta <= 0:
            p.update(player_alive, color, shadow_group)
            shadow_group.update()
            ball_group.update()
            dot_group.update()
            particle_group.update()
            score_msg.update(score)
```

```
            for dot in dot_group:
                if dot.rect.colliderect(p):
                    dot.kill()
                    score_fx.play()
```

```
                    score += 1
                    if highscore <= score:
                        highscore = score
```

```
            if pygame.sprite.spritecollide(p, ball_group, False) and player_alive:
                death_fx.play()
                x, y = p.rect.center
                for i in range(20):
                    particle = Particle(x, y, WHITE, win)
                    particle_group.add(particle)
                player_alive = False
                p.reset()
```

```
            if p.can_move and len(dot_group) == 0 and player_alive:
                dot_index = random.randint(1,4)
                dot_pos = dot_list[dot_index-1]
                dot = Dot(*dot_pos, win)
                dot_group.add(dot)
```

```
                shadow_group.empty()
                shadow = Shadow(dot_index, win)
                shadow_group.add(shadow)
```

```
            if not player_alive and len(particle_group) == 0:
                game_page = False
                score_page = True
```

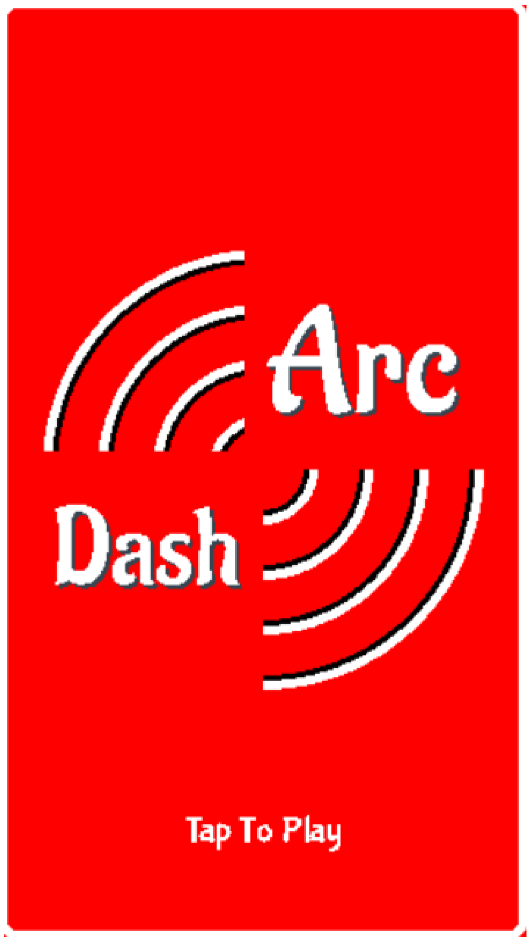
```
                dot_group.empty()
                shadow_group.empty()
                for ball in ball_group:
                    ball.reset()
                score_page_fx.play()
```

```
    pygame.draw.rect(win, WHITE, (0, 0, WIDTH, HEIGHT), 5, border_radius=10)
    clock.tick(FPS)
    pygame.display.update()
```

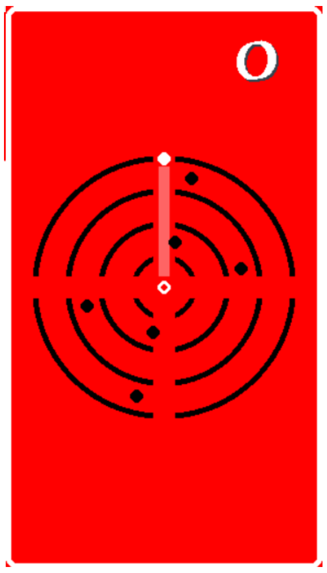
```
pygame.quit()
```

OUTPUT

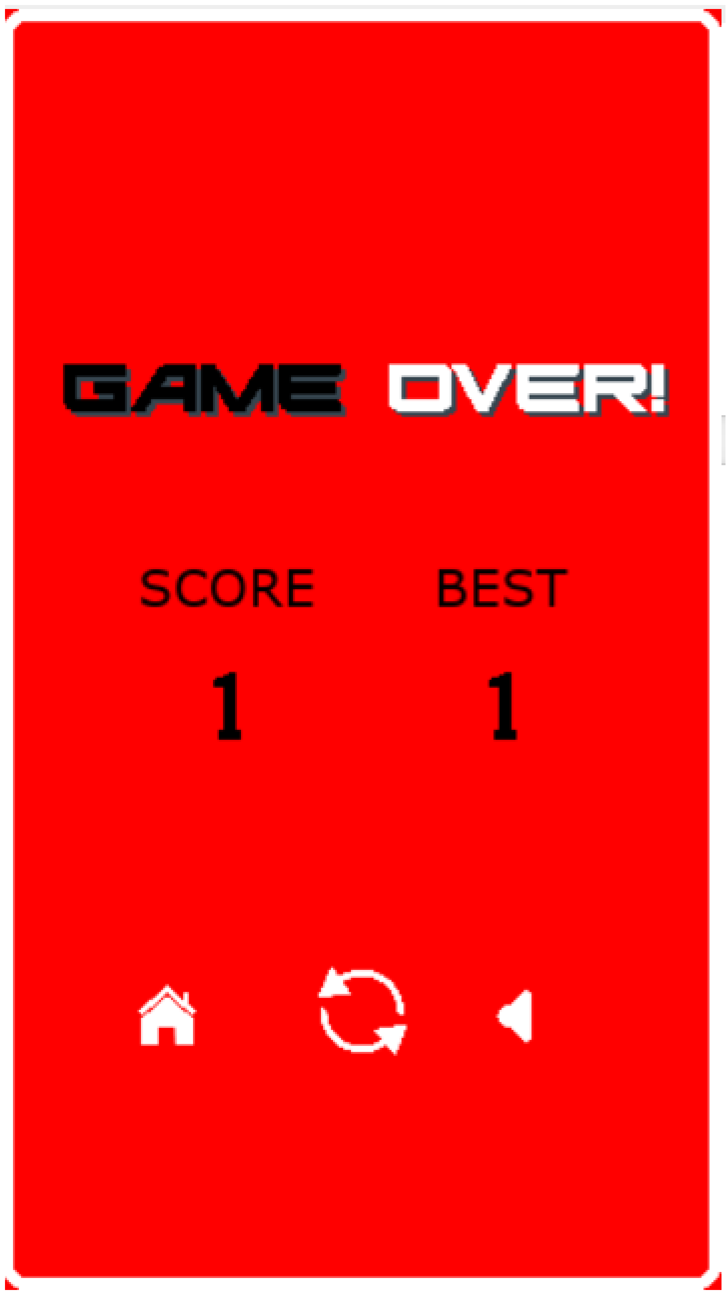
TITLE CARD



GAME PAGE:



END GAME:



GAME OVER!

SCORE BEST

1 1