Creating a 8 Pool Ball By Using Pygames.

```python
import pygame
import pymunk
import pymunk.pygame_util
import math

pygame.init()

SCREEN_WIDTH = 1200
SCREEN_HEIGHT = 678
BOTTOM_PANEL = 50

#game window
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT + BOTTOM_PANEL))
pygame.display.set_caption("Pool")

#pymunk space
space = pymunk.Space()
static_body = space.static_body
draw_options = pymunk.pygame_util.DrawOptions(screen)

#clock
clock = pygame.time.Clock()
FPS = 120

#game variables
dia = 36
pocket_dia = 66
force = 0
max_force = 10000
force_direction = 1
game_running = True
cue_ball_potted = False
taking_shot = True
powering_up = False
potted_balls = []

#colours
BG = (50, 50, 50)
RED = (255, 0, 0)
WHITE = (255, 255, 255)
lives = 3

#fonts
```

```python
font = pygame.font.SysFont("Lato", 30)
large_font = pygame.font.SysFont("Lato", 60)

#load images
cue_image = pygame.image.load("assets/images/cue.png").convert_alpha()
table_image = pygame.image.load("assets/images/table.png").convert_alpha()
ball_images = []
for i in range(1, 17):
  ball_image = pygame.image.load(f"assets/images/ball_{i}.png").convert_alpha()
  ball_images.append(ball_image)

#function for outputting text onto the screen
def draw_text(text, font, text_col, x, y):
  img = font.render(text, True, text_col)
  screen.blit(img, (x, y))

#function for creating balls
def create_ball(radius, pos):
  body = pymunk.Body()
  body.position = pos
  shape = pymunk.Circle(body, radius)
  shape.mass = 5
  shape.elasticity = 0.8
  #use pivot joint to add friction
  pivot = pymunk.PivotJoint(static_body, body, (0, 0), (0, 0))
  pivot.max_bias = 0 # disable joint correction
  pivot.max_force = 1000 # emulate linear friction

  space.add(body, shape, pivot)
  return shape

#setup game balls
balls = []
rows = 5
#potting balls
for col in range(5):
  for row in range(rows):
    pos = (250 + (col * (dia + 1)), 267 + (row * (dia + 1)) + (col * dia / 2))
    new_ball = create_ball(dia / 2, pos)
    balls.append(new_ball)
  rows -= 1
#cue ball
pos = (888, SCREEN_HEIGHT / 2)
cue_ball = create_ball(dia / 2, pos)
balls.append(cue_ball)
```

```
#create six pockets on table
pockets = [
  (55, 63),
  (592, 48),
  (1134, 64),
  (55, 616),
  (592, 629),
  (1134, 616)
]

#create pool table cushions
cushions = [
  [(88, 56), (109, 77), (555, 77), (564, 56)],
  [(621, 56), (630, 77), (1081, 77), (1102, 56)],
  [(89, 621), (110, 600),(556, 600), (564, 621)],
  [(622, 621), (630, 600), (1081, 600), (1102, 621)],
  [(56, 96), (77, 117), (77, 560), (56, 581)],
  [(1143, 96), (1122, 117), (1122, 560), (1143, 581)]
]

#function for creating cushions
def create_cushion(poly_dims):
  body = pymunk.Body(body_type = pymunk.Body.STATIC)
  body.position = ((0, 0))
  shape = pymunk.Poly(body, poly_dims)
  shape.elasticity = 0.8
  space.add(body, shape)

for c in cushions:
  create_cushion(c)

#create pool cue
class Cue():
  def __init__(self, pos):
    self.original_image = cue_image
    self.angle = 0
    self.image = pygame.transform.rotate(self.original_image, self.angle)
    self.rect = self.image.get_rect()
    self.rect.center = pos

  def update(self, angle):
    self.angle = angle

  def draw(self, surface):
    self.image = pygame.transform.rotate(self.original_image, self.angle)
    surface.blit(self.image,
      (self.rect.centerx - self.image.get_width() / 2,
      self.rect.centery - self.image.get_height() / 2)
     )

cue = Cue(balls[-1].body.position)

#create power bars to show how hard the cue ball will be hit
power_bar = pygame.Surface((10, 20))
power_bar.fill(RED)

#game loop
run = True
while run:

  clock.tick(FPS)
  space.step(1 / FPS)

  #fill background
  screen.fill(BG)

  #draw pool table
  screen.blit(table_image, (0, 0))

  #check if any balls have been potted
  for i, ball in enumerate(balls):
    for pocket in pockets:
      ball_x_dist = abs(ball.body.position[0] - pocket[0])
      ball_y_dist = abs(ball.body.position[1] - pocket[1])
      ball_dist = math.sqrt((ball_x_dist ** 2) + (ball_y_dist ** 2))
      if ball_dist <= pocket_dia / 2:
        #check if the potted ball was the cue ball
        if i == len(balls) - 1:
          lives -= 1
          cue_ball_potted = True
          ball.body.position = (-100, -100)
          ball.body.velocity = (0.0, 0.0)
        else:
          space.remove(ball.body)
          balls.remove(ball)
          potted_balls.append(ball_images[i])
          ball_images.pop(i)

  #draw pool balls
  for i, ball in enumerate(balls):
    screen.blit(ball_images[i], (ball.body.position[0] - ball.radius,
ball.body.position[1] - ball.radius))

  #check if all the balls have stopped moving
  taking_shot = True
  for ball in balls:
    if int(ball.body.velocity[0]) != 0 or int(ball.body.velocity[1]) != 0:
      taking_shot = False

  #draw pool cue
  if taking_shot == True and game_running == True:
    if cue_ball_potted == True:
      #reposition cue ball
      balls[-1].body.position = (888, SCREEN_HEIGHT / 2)
      cue_ball_potted = False
    #calculate pool cue angle
    mouse_pos = pygame.mouse.get_pos()
    cue.rect.center = balls[-1].body.position
    x_dist = balls[-1].body.position[0] - mouse_pos[0]
    y_dist = -(balls[-1].body.position[1] - mouse_pos[1]) # -ve because pygame y
coordinates increase down the screen
    cue_angle = math.degrees(math.atan2(y_dist, x_dist))
    cue.update(cue_angle)
    cue.draw(screen)

  #power up pool cue
  if powering_up == True and game_running == True:
    force += 100 * force_direction
    if force >= max_force or force <= 0:
      force_direction *= -1
    #draw power bars
    for b in range(math.ceil(force / 2000)):
      screen.blit(power_bar,
       (balls[-1].body.position[0] - 30 + (b * 15),
        balls[-1].body.position[1] + 30))
  elif powering_up == False and taking_shot == True:
    x_impulse = math.cos(math.radians(cue_angle))
    y_impulse = math.sin(math.radians(cue_angle))
    balls[-1].body.apply_impulse_at_local_point((force * -x_impulse, force *
y_impulse), (0, 0))
    force = 0
    force_direction = 1

  #draw bottom panel
  pygame.draw.rect(screen, BG, (0, SCREEN_HEIGHT, SCREEN_WIDTH, BOTTOM_PANEL))
  draw_text("LIVES: " + str(lives), font, WHITE, SCREEN_WIDTH - 200,
SCREEN_HEIGHT + 10)

  #draw potted balls in bottom panel
  for i, ball in enumerate(potted_balls):
    screen.blit(ball, (10 + (i * 50), SCREEN_HEIGHT + 10))

  #check for game over
  if lives <= 0:
    draw_text("GAME OVER", large_font, WHITE, SCREEN_WIDTH / 2 - 160,
SCREEN_HEIGHT / 2 - 100)
    game_running = False

  #check if all balls are potted
  if len(balls) == 1:
    draw_text("YOU WIN!", large_font, WHITE, SCREEN_WIDTH / 2 - 160,
SCREEN_HEIGHT / 2 - 100)
    game_running = False

  #event handler
  for event in pygame.event.get():
    if event.type == pygame.MOUSEBUTTONDOWN and taking_shot == True:
      powering_up = True
    if event.type == pygame.MOUSEBUTTONUP and taking_shot == True:
      powering_up = False
    if event.type == pygame.QUIT:
      run = False

  #space.debug_draw(draw_options)
  pygame.display.update()

pygame.quit()
```