Bing

Hallo, das ist Bing. Ich kann versuchen, Python-Softwarecode für Ihre Spielidee zu erstellen. Bitte beachten Sie, dass der Code möglicherweise nicht perfekt ist und einige Fehler oder Verbesserungsmöglichkeiten aufweisen kann. Ich hoffe, Sie haben Spaß mit Ihrem Naschschatz-Abenteuer im Dunkeln. Hier ist der Code, den ich generiert habe:

```
# Naschschatz-Abenteuer im Dunkeln
# Ein Spiel, in dem man im Dunkeln nach Schätzen sucht und Punkte sammelt
# Benötigt die Pygame-Bibliothek

import pygame
import random

# Konstanten für die Fenstergröße
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600

# Konstanten für die Farben
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
YELLOW = (255, 255, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

# Konstanten für die Spielobjekte
PLAYER_SIZE = 20
PLAYER_SPEED = 5
WALL_SIZE = 10
WALL_COLOR = WHITE
LAMP_SIZE = 30
LAMP_COLOR = YELLOW
LAMP_RADIUS = 100
COIN_SIZE = 10
COIN_COLOR = YELLOW
COIN_VALUE = 10
CANDY_SIZE = 10
CANDY_COLOR = RED
CANDY_VALUE = 1
MAX_LIVES = 4
MAX_POINTS = 1000
MAX_LEVELS = 50
MAX_TIME = 120 # in Sekunden

# Initialisierung von Pygame
pygame.init()
window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("Naschschatz-Abenteuer im Dunkeln")
clock = pygame.time.Clock()

# Erstellung einer Klasse für den Spieler
class Player:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.rect = pygame.Rect(self.x, self.y, PLAYER_SIZE, PLAYER_SIZE)
        self.color = GREEN
        self.lives = MAX_LIVES
        self.points = 0
        self.level = 1

    def draw(self):
        pygame.draw.rect(window, self.color, self.rect)

    def move(self, dx, dy):
        self.x += dx
        self.y += dy
        self.rect = pygame.Rect(self.x, self.y, PLAYER_SIZE, PLAYER_SIZE)

    def collide(self, other):
        return self.rect.colliderect(other.rect)

# Erstellung einer Klasse für die Wände
class Wall:
    def __init__(self, x, y, width, height):
        self.x = x
        self.y = y
        self.rect = pygame.Rect(self.x, self.y, width, height)
```

```
        self.color = WALL_COLOR

    def draw(self):
        pygame.draw.rect(window, self.color, self.rect)

# Erstellung einer Klasse für die Lampen
class Lamp:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.rect = pygame.Rect(self.x, self.y, LAMP_SIZE, LAMP_SIZE)
        self.color = LAMP_COLOR
        self.on = False

    def draw(self):
        pygame.draw.rect(window, self.color, self.rect)
        if self.on:
            pygame.draw.circle(window, self.color, (self.x + LAMP_SIZE // 2, self.y + LAMP_SIZE // 2), LAMP_RADIUS, 1)

    def toggle(self):
        self.on = not self.on

    def illuminate(self, other):
        return self.on and ((self.x - other.x) ** 2 + (self.y - other.y) ** 2) <= (LAMP_RADIUS + other.size // 2) ** 2

# Erstellung einer Klasse für die Münzen
class Coin:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.rect = pygame.Rect(self.x, self.y, COIN_SIZE, COIN_SIZE)
        self.color = COIN_COLOR
        self.value = COIN_VALUE
        self.size = COIN_SIZE

    def draw(self):
        pygame.draw.rect(window, self.color, self.rect)

# Erstellung einer Klasse für die Süßigkeiten
class Candy:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.rect = pygame.Rect(self.x, self.y, CANDY_SIZE, CANDY_SIZE)
        self.color = CANDY_COLOR
        self.value = CANDY_VALUE
        self.size = CANDY_SIZE

    def draw(self):
        pygame.draw.rect(window, self.color, self.rect)

# Erstellung einer Funktion für die Textanzeige
def draw_text(text, x, y, size, color):
    font = pygame.font.SysFont("Arial", size)
    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect()
    text_rect.center = (x, y)
    window.blit(text_surface, text_rect)

# Erstellung einer Funktion für die Spiellogik
def game_logic():
    global running, player, walls, lamps, coins, candies, time_left, game_over, game_win

    # Abfrage der Tastatureingaben
    keys = pygame.key.get_pressed()
    dx = 0
    dy = 0
    if keys[pygame.K_LEFT]:
        dx = -PLAYER_SPEED
    if keys[pygame.K_RIGHT]:
        dx = PLAYER_SPEED
    if keys[pygame.K_UP]:
        dy = -PLAYER_SPEED
    if keys[pygame.K_DOWN]:
        dy = PLAYER_SPEED

    # Bewegung des Spielers
    player.move(dx, dy)

    # Kollision mit den Wänden
    for wall in walls:
        if player.collide(wall):
            player.lives -= 1
```

```
            player.x -= dx
            player.y -= dy
            player.rect = pygame.Rect(player.x, player.y, PLAYER_SIZE, PLAYER_SIZE)
            break

    # Kollision mit den Lampen
    for lamp in lamps:
        if player.collide(lamp):
            lamp.toggle()

    # Kollision mit den Münzen
    for coin in coins:
        if player.collide(coin):
            player.points += coin.value
            coins.remove(coin)

    # Kollision mit den Süßigkeiten
    for candy in candies:
        if player.collide(candy):
            player.lives = min(player.lives + candy.value, MAX_LIVES)
            candies.remove(candy)

    # Überprüfung der Spielbedingungen
    if player.lives <= 0:
        game_over = True
    elif player.points >= MAX_POINTS:
        game_win = True
    elif time_left <= 0:
        game_over = True
    elif len(coins) == 0 and len(candies) == 0:
        player.level += 1
        generate_level()

    # Aktualisierung der verbleibenden Zeit
    time_left -= clock.get_time() / 1000

# Erstellung einer Funktion für die Grafikausgabe
def game_draw():
    global running, player, walls, lamps, coins, candies, time_left, game_over, game_win

    # Füllen des Hintergrunds mit Schwarz
    window.fill(BLACK)

    # Zeichnen der Spielobjekte
    player.draw()
    for wall in walls:
        if any(lamp.illuminate(wall) for lamp in lamps):
            wall.draw()
    for lamp in lamps:
        lamp.draw()
    for coin in coins:
        if any(lamp.illuminate(coin) for lamp in lamps):
            coin.draw()
    for candy in candies:
        if any(lamp.illuminate(candy) for lamp in lamps):
            candy.draw()

    # Zeichnen der Spielinformationen
    draw_text(f"Leben: {player.lives}", 50, 20, 20, WHITE)
    draw_text(f"Punkte: {player.points}", 150, 20, 20, WHITE)
    draw_text(f"Level: {player.level}", 250, 20, 20, WHITE)
    draw_text(f"Zeit: {int(time_left)}", 350, 20, 20, WHITE)

    # Zeichnen der Spielendenachrichten
    if game_over:
        draw_text("Game Over", WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2, 40, RED)
    elif game_win:
        draw_text("Du hast gewonnen!", WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2, 40, GREEN)

    # Aktualisierung des Fensters
    pygame.display.update()

# Erstellung einer Funktion für die Levelgenerierung
def generate_level():
    global running, player, walls, lamps, coins, candies, time_left, game_over, game_win

    # Zurücksetzen der Spielobjekte
    walls = []
    lamps = []
    coins = []
    candies = []

    # Erstellung der Außenwände
```

```
walls.append(Wall(0, 0, WINDOW_WIDTH, WALL_SIZE))
walls.append(Wall(0, 0, WALL_SIZE, WINDOW_HEIGHT))
walls.append(Wall(0, WINDOW_HEIGHT - WALL_SIZE, WINDOW_WIDTH, WALL_SIZE))
walls.append(Wall(W
```