```
# Toadie
# Code Angel

# Classes: Toad

import pygame

import toadie
import world

# Define constants
TOAD_START_X = 6
TOAD_START_Y = world.PAVEMENT_LANE_1

TOAD_DEATH_TIME = 2


class Toad:

    def __init__(self):
        self.image = toadie.load_media('image', 'toad')
        self.dead_image = toadie.load_media('image', 'dead_toad')

        self.hop_sound = toadie.load_media('audio', 'hop')
        self.death_sound = toadie.load_media('audio', 'death')
        self.home_sound = toadie.load_media('audio', 'home')

        self.rect = self.image.get_rect()
        self.padding_width = (world.BLOCK_SIZE - self.image.get_width()) / 2
        self.padding_height = (world.BLOCK_SIZE - self.image.get_height()) / 2

        self.rect.x = TOAD_START_X * world.BLOCK_SIZE + self.padding_width
        self.rect.y = TOAD_START_Y * world.BLOCK_SIZE + self.padding_height

        self.lives = 3
        self.points = 0
        self.home_count = 0
        self.alive = True
        self.death_pause_timer = 0
        self.furthest_forward = self.rect.y

```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
    # Draw toad either dead or alive)
    def draw(self, game_screen):
        if self.alive is True:
            game_screen.blit(self.image, [self.rect.x, self.rect.y])
        else:
            game_screen.blit(self.dead_image, [self.rect.x, self.rect.y])

    # Move toad one block if alive and not at edge of screen
    def move(self, direction):

        if self.alive is True:

            # Right
            if direction == 'R':
                if self.rect.x + world.BLOCK_SIZE < toadie.SCREEN_WIDTH:
                    self.rect.x = self.rect.x + world.BLOCK_SIZE
                    self.hop_sound.play()

            # Left
            elif direction == 'L':
                if self.rect.x - world.BLOCK_SIZE > 0:
                    self.rect.x = self.rect.x - world.BLOCK_SIZE
                    self.hop_sound.play()

            # Up
            elif direction == 'U':
                self.rect.y = self.rect.y - world.BLOCK_SIZE
                self.hop_sound.play()

                # 10 points awarded each step toadie takes towards home
                # If he goes down and then up again, no extra points are awarded
                # furthest_forward tracks the furthest toadie has been up the screen
                if self.rect.y < self.furthest_forward:
                    self.furthest_forward = self.rect.y
                    self.points += 10

            # Down
            elif direction == 'D':
                if self.rect.y + 3 * world.BLOCK_SIZE < toadie.SCREEN_HEIGHT:
                    self.rect.y = self.rect.y + world.BLOCK_SIZE
                    self.hop_sound.play()
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

    # Check if toadie has collided with a vehicle
    def check_collision(self, vehicle_list):
        for vehicle in vehicle_list:
            if self.rect.colliderect(vehicle.rect):
                self.die()

    # Check if toadie has collided with a river object
    def check_water(self, river_list):

        toad_top = self.calc_toad_top()
        river_bottom_edge = world.PAVEMENT_LANE_2 * world.BLOCK_SIZE
        river_top_edge = world.HOME_LANE * world.BLOCK_SIZE

        # Is toadie between the middle pavement and home - if so that is the river
        if river_top_edge < toad_top < river_bottom_edge:

            floating_toad = False

            # Loop through the river list checking if toadie has collided with an item and so will float
            for river_item in river_list:
                if self.rect.colliderect(river_item):
                    if self.rect.left > river_item.rect.left and self.rect.right < river_item.rect.right:

                        # If he has, floating_toad is set to True because he will float
                        floating_toad = True

                        # Toadie will move horizontally at the same speed as the river object he is floating on
                        self.rect.x = self.rect.x + river_item.speed

            # Toadie is on the river, but not floating on a log or a turtle, so he dies
            if floating_toad is False:
                self.die()

            # Toadie is on a floating object but he has floated off the right of the screen so he dies
            if floating_toad is True and self.rect.right >= toadie.SCREEN_WIDTH:
                self.die()

            # Toadie is on a floating object but he has floated off the left of the screen so he dies
            if floating_toad is True and self.rect.left <= 0:
                self.die()
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0   |1   |2   |3   |4   |5   |6   |7   |8

    # Check if Toadie has reached a home pad
    def check_home(self, home_pads, game_timer):

        toad_top = self.calc_toad_top()
        home_loc = world.HOME_LANE * world.BLOCK_SIZE

        # Is toadie in the same row as the the home pads?
        if toad_top <= home_loc:

            found_home = False

            # Loop through each pad
            for pad in home_pads:

                # Check if Toadie is in the pad (his horizontal centre must be within the pad)
                if pad.rect.left <= self.rect.centerx <= pad.rect.right:

                    # Check that the pad is not already occupied
                    if pad.occupied is False:
                        found_home = True

                        # Set the occupied property of the pad that was found to True so it cannot be occupied again
                        pad.occupied = True

                        # Play home sound
                        self.home_sound.play()

                        # Update the score: 50 points for getting a home pad
                        self.points += 50
                        self.home_count += 1

                        # Update the score: 1000 bonus points for filling all 5 home pads
                        if self.home_count == 5:
                            self.points += 1000

                        # Update the score: 10 points for each second left on the clock
                        secs_left = game_timer.get_seconds_left()
                        self.points += 10 * secs_left

                        # Reset Toadie location and timer
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                        self.rect.x = TOAD_START_X * world.BLOCK_SIZE + self.padding_width
                        self.rect.y = TOAD_START_Y * world.BLOCK_SIZE + self.padding_height

                        self.furthest_forward = self.rect.y
                        game_timer.reset()

                    # The pad was occupied so Toadie dies
                    else:
                        self.die()

            # Toadie is in the same row as the home pads, but he missed them all and so he dies
            if found_home is False:
                self.die()

    # Toadie has died
    def die(self):
        if self.alive is True:
            self.alive = False

            # Lose a life
            self.lives -= 1

            # Play death sound effect
            self.death_sound.play()

            # Start the death pause timer
            if self.lives > 0:
                self.death_pause_timer = pygame.time.get_ticks()

    # Check to see if there is time on the death pause timer
    def check_death_pause(self, game_timer):

        # Calculate the time in ticks since the timer started
        elapsed_time = pygame.time.get_ticks() - self.death_pause_timer

        # If the timer has gone beyond the length of time the toad skeleton should be displayed,
        if elapsed_time > TOAD_DEATH_TIME * world.MILLISECONDS:

            # Reset toadie and the timer
            self.alive = True
            self.rect.x = TOAD_START_X * world.BLOCK_SIZE + self.padding_width
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
            self.rect.y = TOAD_START_Y * world.BLOCK_SIZE + self.padding_height
            self.furthest_forward = self.rect.y
            game_timer.reset()

    # Used to update the score - once points have been collected (added to the score) they are reset to 0
    def collect_points(self):
        collected_points = self.points
        self.points = 0

        return collected_points

    # Calculate toadie's y coordinate
    def calc_toad_top(self):
        return self.rect.y - self.padding_height
|0  |1  |2  |3  |4  |5  |6  |7  |8
```