

Pacman Reverse

Thursday, December 28, 2023 10:38 AM

At this point, Pacman can now reverse direction while he's moving in between nodes

Pacman.py

```
import pygame
from pygame.locals import *
from vector import Vector2
from constants import *

class Pacman(object):
    def __init__(self, node):
        self.name = PACMAN
        self.position = Vector2(200, 400)
        self.directions = {STOP:Vector2(), UP:Vector2(0,-1), DOWN:Vector2(0,1), LEFT:Vector2(-1,0),
RIGHT:Vector2(1,0)}
        self.direction = STOP
        self.speed = 100
        self.radius = 10
        self.color = YELLOW
        self.node = node
        self.setPosition()
        self.target = node

    def setPosition(self):
        self.position = self.node.position.copy()

    def update(self, dt):
        self.position += self.directions[self.direction]*self.speed*dt
        direction = self.getValidKey()
        if self.overshotTarget():
            self.node = self.target
            self.target = self.getNewTarget(direction)
            if self.target is not self.node:
                self.direction = direction
            else:
                self.target = self.getNewTarget(self.direction)

            if self.target is self.node:
                self.direction = STOP
                self.setPosition()
            else:
                if self.oppositeDirection(direction):
                    self.reverseDirection()

    def reverseDirection(self):
```

```

self.direction *= -1
temp = self.node
self.node = self.target
self.target = temp

def oppositeDirection(self, direction):
    if direction is not STOP:
        if direction == self.direction * -1:
            return True
    return False

def overshotTarget(self):
    if self.target is not None:
        vec1 = self.target.position - self.node.position
        vec2 = self.position - self.node.position
        node2Target = vec1.magnitudeSquared()
        node2Self = vec2.magnitudeSquared()
        return node2Self >= node2Target
    return False

def validDirection(self, direction):
    if direction is not STOP:
        if self.node.neighbors[direction] is not None:
            return True
    return False

def getNewTarget(self, direction):
    if self.validDirection(direction):
        return self.node.neighbors[direction]
    return self.node

def getValidKey(self):
    key_pressed = pygame.key.get_pressed()
    if key_pressed[K_UP]:
        return UP
    if key_pressed[K_DOWN]:
        return DOWN
    if key_pressed[K_LEFT]:
        return LEFT
    if key_pressed[K_RIGHT]:
        return RIGHT
    return STOP

def render(self, screen):
    p = self.position.asInt()
    pygame.draw.circle(screen, self.color, p, self.radius)

```