

Erasing Sketch Lines

The sketch lines are now gone, leaving us with a pristine maze graphic. Our only flaw is the characters look kinda off-center. The last 2 additions we made were supposed to fix that, but it's still off. This is another issue I'm gonna let stay.

Run.py

```
import pygame
from pygame.locals import *
from constants import *
from pacman import Pacman
from nodes import NodeGroup
from pellets import PelletGroup
from ghosts import GhostGroup
from fruit import Fruit
from pauser import Pause
from text import TextGroup
from sprites import LifeSprites
from sprites import MazeSprites

class GameController(object):
    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode(SCREENSIZE, 0, 32)
        self.background = None
        self.clock = pygame.time.Clock()
        self.fruit = None
        self.pause = Pause(True)
        self.level = 0
        self.lives = 5
        self.score = 0
        self.textgroup = TextGroup()
        self.lifesprites = LifeSprites(self.lives)

    def setBackground(self):
        self.background = pygame.surface.Surface(SCREENSIZE).convert()
        self.background.fill(BLACK)

    def startGame(self):
        self.setBackground()
        self.mazesprites = MazeSprites("maze01.txt", "maze_rotation1.txt")
        self.background = self.mazesprites.constructBackground(self.background, self.level%5)
        self.nodes = NodeGroup("maze01.txt")
        self.nodes.setPortalPair((0,17), (27,17))
        homekey = self.nodes.createHomeNodes(11.5, 14)
        self.nodes.connectHomeNodes(homekey, (12,14), LEFT)
        self.nodes.connectHomeNodes(homekey, (15,14), RIGHT)
        self.pacman = Pacman(self.nodes.getNodeFromTiles(15, 26))
```

```

self.pellets = PelletGroup("maze01.txt.")
self.ghosts = GhostGroup(self.nodes.getStartTempNode(), self.pacman)
self.ghosts.blinky.setStartNode(self.nodes.getNodeFromTiles(2+11.5, 0+14))
self.ghosts.pinky.setStartNode(self.nodes.getNodeFromTiles(2+11.5, 3+14))
self.ghosts.inky.setStartNode(self.nodes.getNodeFromTiles(0+11.5, 3+14))
self.ghosts.clyde.setStartNode(self.nodes.getNodeFromTiles(4+11.5, 3+14))
self.ghosts.setSpawnNode(self.nodes.getNodeFromTiles(2+11.5, 3+14))
self.nodes.denyHomeAccess(self.pacman)
self.nodes.denyHomeAccessList(self.ghosts)
self.nodes.denyAccessList(2+11.5, 3+14, LEFT, self.ghosts)
self.nodes.denyAccessList(2+11.5, 3+14, RIGHT, self.ghosts)
self.ghosts.inky.startNode.denyAccess(RIGHT, self.ghosts.inky)
self.ghosts.clyde.startNode.denyAccess(LEFT, self.ghosts.clyde)
self.nodes.denyAccessList(12, 14, UP, self.ghosts)
self.nodes.denyAccessList(15, 14, UP, self.ghosts)
self.nodes.denyAccessList(12, 26, UP, self.ghosts)
self.nodes.denyAccessList(15, 26, UP, self.ghosts)

def update(self):
    dt = self.clock.tick(30) / 1000.0
    self.textgroup.update(dt)
    self.pellets.update(dt)
    if not self.pause.paused:
        self.pacman.update(dt)
        self.ghosts.update(dt)
        if self.fruit is not None:
            self.fruit.update(dt)
        self.checkGhostEvents()
        self.checkPelletEvents()
        self.checkFruitEvents
    afterPauseMethod = self.pause.update(dt)
    if afterPauseMethod is not None:
        afterPauseMethod()
    self.checkEvents()
    self.render()

def updateScore(self, points):
    self.score += points
    self.textgroup.updateScore(self.score)

def checkEvents(self):
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        elif event.type == KEYDOWN:
            if event.key == K_SPACE:
                if self.pacman.alive:
                    self.pause.setPause(playerPaused=True)
                    if not self.pause.paused:
                        self.textgroup.hideText()
                        self.showEntities()
                    else:
                        self.hideEntities()

```

```

        self.textgroup.showText(PAUSETXT)

def checkGhostEvents(self):
    for ghost in self.ghosts:
        if self.pacman.collideGhost(ghost):
            if ghost.mode.current is FREIGHT:
                self.pacman.visible = False
                ghost.visible = False
                self.updateScore(ghost.points)
                self.textgroup.addText(str(ghost.points), WHITE, ghost.position.x, ghost.position.y, 8, time=1)
                self.ghosts.updatePoints()
                self.pause.setPause(pauseTime=1, func=self.showEntities)
                ghost.startSpawn()
                self.nodes.allowHomeAccess(ghost)
            elif ghost.mode.current is not SPAWN:
                if self.pacman.alive:
                    self.lives -= 1
                    self.lifesprites.removeImage()
                    self.pacman.die()
                    self.ghosts.hide()
                    if self.lives <= 0:
                        self.textgroup.showText(GAMEOVERTXT)
                        self.pause.setPause(pauseTime=3, func=self.restartGame)
                    else:
                        self.pause.setPause(pauseTime=3, func=self.resetLevel)

def checkPelletEvents(self):
    pellet = self.pacman.eatPellets(self.pellets.pelletList)
    if pellet:
        self.pellets.numEaten += 1
        self.updateScore(pellet.points)
        if self.pellets.numEaten == 30:
            self.ghosts.inky.startNode.allowAccess(RIGHT, self.ghosts.inky)
        if self.pellets.numEaten == 70:
            self.ghosts.clyde.startNode.allowAccess(LEFT, self.ghosts.clyde)
        self.pellets.pelletList.remove(pellet)
        if pellet.name == POWERPELLET:
            self.ghosts.startFreight()
        if self.pellets.isEmpty():
            self.hideEntities()
            self.pause.setPause(pauseTime=3, func=self.nextLevel)

def checkFruitEvents(self):
    if self.pellets.numEaten == 50 or self.pellets.numEaten == 140:
        if self.fruit is None:
            self.fruit = Fruit(self.nodes.getNodeFromTiles(9, 20))
        if self.fruit is not None:
            if self.pacman.collideCheck(self.fruit):
                self.updateScore(self.fruit.points)
                self.textgroup.addText(str(self.fruit.points), WHITE, self.fruit.position.x, self.fruit.position.y, 8,
time=1)
                self.fruit = None
            elif self.fruit.destroy:

```

```

        self.fruit = None

    def showEntities(self):
        self.pacman.visible = True
        self.ghosts.show()

    def hideEntities(self):
        self.pacman.visible = False
        self.ghosts.hide()

    def nextLevel(self):
        self.showEntities()
        self.level += 1
        self.pause.paused = True
        self.textgroup.updateLevel(self.level)
        self.startGame()

    def restartGame(self):
        self.lives = 5
        self.level = 0
        self.pause.paused = True
        self.fruit = None
        self.score = 0
        self.textgroup.updateScore(self.score)
        self.textgroup.updateLevel(self.level)
        self.textgroup.showText(READYTXT)
        self.lifesprites.resetLives(self.lives)
        self.startGame()

    def resetLevel(self):
        self.pause.paused = True
        self.pacman.reset()
        self.ghosts.reset()
        self.fruit = None
        self.textgroup.showText(READYTXT)

    def render(self):
        self.screen.blit(self.background, (0,0))
        self.pellets.render(self.screen)
        if self.fruit is not None:
            self.fruit.render(self.screen)
        self.pacman.render(self.screen)
        self.ghosts.render(self.screen)
        self.textgroup.render(self.screen)

        for i in range(len(self.lifesprites.images)):
            x = self.lifesprites.images[i].get_width() * i
            y = SCREENHEIGHT - self.lifesprites.images[i].get_height()
            self.screen.blit(self.lifesprites.images[i], (x, y))

    pygame.display.update()

```

```

if __name__ == "__main__":
    game = GameController()
    game.startGame()
    while True:
        game.update()

```

Entity.py

```

import pygame
from pygame.locals import *
from vector import Vector2
from constants import *
from random import randint

class Entity(object):
    def __init__(self, node):
        self.name = None
        self.directions = {UP:Vector2(0, -1),DOWN:Vector2(0, 1),
                           LEFT:Vector2(-1, 0), RIGHT:Vector2(1, 0), STOP:Vector2()}
        self.direction = STOP
        self.setSpeed(100)
        self.radius = 10
        self.collideRadius = 5
        self.color = WHITE
        self.visible = True
        self.disablePortal = False
        self.goal = None
        self.directionMethod = self.randomDirection
        self.setStartNode(node)
        self.image = None

    def setStartNode(self, node):
        self.node = node
        self.startNode = node
        self.target = node
        self.setPosition()

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

    def validDirection(self, direction):
        if direction is not STOP:
            if self.name in self.node.access[direction]:
                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 overshootTarget(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 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 setSpeed(self, speed):
    self.speed = speed * TILEWIDTH / 16

def render(self, screen):
    if self.visible:
        if self.image is not None:
            adjust = Vector2(TILEWIDTH, TILEHEIGHT)
            p = self.position - adjust
            screen.blit(self.image, p.asTuple())
        else:
            p = self.position.asInt()
            pygame.draw.circle(screen, self.color, p, self.radius)

def update(self, dt):
    self.position += self.directions[self.direction]*self.speed*dt

    if self.overshootTarget():
        self.node = self.target
        directions = self.validDirections()
        direction = self.directionMethod(directions)
        if not self.disablePortal:
            if self.node.neighbors[PORTAL] is not None:
                self.node = self.node.neighbors[PORTAL]
        self.target = self.getNewTarget(direction)
        if self.target is not self.node:
            self.direction = direction
        else:
            self.target = self.getNewTarget(self.direction)

    self.setPosition()

```

```

def validDirections(self):
    directions = []
    for key in [UP, DOWN, LEFT, RIGHT]:
        if self.validDirection(key):
            if key != self.direction * -1:
                directions.append(key)
    if len(directions) == 0:
        directions.append(self.direction * -1)
    return directions

def randomDirection(self, directions):
    return directions[randint(0, len(directions)-1)]

def goalDirection(self, directions):
    distances = []
    for direction in directions:
        vec = self.node.position + self.directions[direction]*TILEWIDTH - self.goal
        distances.append(vec.magnitudeSquared())
    index = distances.index(min(distances))
    return directions[index]

def setBetweenNodes(self, direction):
    if self.node.neighbors[direction] is not None:
        self.target = self.node.neighbors[direction]
        self.position = (self.node.position + self.target.position) / 2.0

def reset(self):
    self.setStartNode(self.startNode)
    self.direction = STOP
    self.speed = 100
    self.visible = True

```

Sprites.py

```

import pygame
from constants import *
import numpy as np

BASETILEWIDTH = 16
BASETILEHEIGHT = 16

class Spritesheet(object):
    def __init__(self):
        self.sheet = pygame.image.load("spritesheet.png").convert()
        transcolor = self.sheet.get_at((0,0))
        self.sheet.set_colorkey(transcolor)
        width = int(self.sheet.get_width() / BASETILEWIDTH * TILEWIDTH)
        height = int(self.sheet.get_height() / BASETILEHEIGHT * TILEHEIGHT)
        self.sheet = pygame.transform.scale(self.sheet, (width, height))

    def getImage(self, x, y, width, height):

```

```

x *= TILEWIDTH
y *= TILEHEIGHT
self.sheet.set_clip(pygame.Rect(x, y, width, height))
return self.sheet.subsurface(self.sheet.get_clip())

class PacmanSprites(SpriteSheet):
    def __init__(self, entity):
        SpriteSheet.__init__(self)
        self.entity = entity
        self.entity.image = self.getStartImage()

    def getStartImage(self):
        return self.getImage(8, 0)

    def getImage(self, x, y):
        return SpriteSheet.getImage(self, x, y, 2*TILEWIDTH, 2*TILEHEIGHT)

class GhostSprites(SpriteSheet):
    def __init__(self, entity):
        SpriteSheet.__init__(self)
        self.x = {BLINKY:0, PINKY:2, INKY:4, CLYDE:6}
        self.entity = entity
        self.entity.image = self.getStartImage()

    def getStartImage(self):
        return self.getImage(self.x[self.entity.name], 4)

    def getImage(self, x, y):
        return SpriteSheet.getImage(self, x, y, 2*TILEWIDTH, 2*TILEHEIGHT)

class FruitSprites(SpriteSheet):
    def __init__(self, entity):
        SpriteSheet.__init__(self)
        self.entity = entity
        self.entity.image = self.getStartImage()

    def getStartImage(self):
        return self.getImage(16, 8)

    def getImage(self, x, y):
        return SpriteSheet.getImage(self, x, y, 2*TILEWIDTH, 2*TILEHEIGHT)

class LifeSprites(SpriteSheet):
    def __init__(self, numlives):
        SpriteSheet.__init__(self)
        self.resetLives(numlives)

    def removeImage(self):
        if len(self.images) > 0:
            self.images.pop(0)

    def resetLives(self, numlives):
        self.images = []

```



```

        for i in range(numlives):
            self.images.append(self.getImage(0,0))

    def getImage(self, x, y):
        return Spritesheet.getImage(self, x, y, 2*TILEWIDTH, 2*TILEHEIGHT)

class MazeSprites(Spritesheet):
    def __init__(self, mazefile, rotfile):
        Spritesheet.__init__(self)
        self.data = self.readMazeFile(mazefile)
        self.rotdata = self.readMazeFile(rotfile)

    def getImage(self, x, y):
        return Spritesheet.getImage(self, x, y, TILEWIDTH, TILEHEIGHT)

    def readMazeFile(self, mazefile):
        return np.loadtxt(mazefile, dtype='<U1')

    def constructBackground(self, background, y):
        for row in list(range(self.data.shape[0])):
            for col in list(range(self.data.shape[1])):
                if self.data[row][col].isdigit():
                    x = int(self.data[row][col]) + 12
                    sprite = self.getImage(x, y)
                    rotval = int(self.rotdata[row][col])
                    sprite = self.rotate(sprite, rotval)
                    background.blit(sprite, (col*TILEWIDTH, row*TILEHEIGHT))
                elif self.data[row][col] == '=':
                    sprite = self.getImage(10, 8)
                    background.blit(sprite, (col*TILEWIDTH, row*TILEHEIGHT))

        return background

    def rotate(self, sprite, value):
        return pygame.transform.rotate(sprite, value*90)

```

Pellets.py

```

import pygame
from vector import Vector2
from constants import *
import numpy as np

class Pellet(object):
    def __init__(self, row, column):
        self.name = PELLET
        self.position = Vector2(column*TILEWIDTH, row*TILEHEIGHT)
        self.color = WHITE
        self.radius = int(2 * TILEWIDTH / 16)
        self.collideRadius = int(2 * TILEWIDTH / 16)
        self.points = 10
        self.visible = True

```

```

def render(self, screen):
    if self.visible:
        adjust = Vector2(TILEWIDTH, TILEHEIGHT) / 2
        p = self.position + adjust
        pygame.draw.circle(screen, self.color, p.asInt(), self.radius)

class PowerPellet(Pellet):
    def __init__(self, row, column):
        Pellet.__init__(self, row, column)
        self.name = POWERPELLET
        self.radius = int(8 * TILEWIDTH / 16)
        self.points = 50
        self.flashTime = 0.2
        self.timer = 0

    def update(self, dt):
        self.timer += dt
        if self.timer >= self.flashTime:
            self.visible = not self.visible
            self.timer = 0

class PelletGroup(object):
    def __init__(self, pelletfile):
        self.pelletList = []
        self.powerpellets = []
        self.createPelletList(pelletfile)
        self.numEaten = 0

    def update(self, dt):
        for powerpellet in self.powerpellets:
            powerpellet.update(dt)

    def createPelletList(self, pelletfile):
        data = self.readPelletfile(pelletfile)
        for row in range(data.shape[0]):
            for col in range(data.shape[1]):
                if data[row][col] in ['. ', '+']:
                    self.pelletList.append(Pellet(row, col))
                elif data[row][col] in ['P', 'p']:
                    pp = PowerPellet(row, col)
                    self.pelletList.append(pp)
                    self.powerpellets.append(pp)

    def readPelletfile(self, textfile):
        return np.loadtxt(textfile, dtype='<U1')

    def isEmpty(self):
        if len(self.pelletList) == 0:
            return True
        return False

    def render(self, screen):

```

```
for pellet in self.pelletList:
    pellet.render(screen)
```

Maze01.txt

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
011111111111117811111111111110
1+....+.....+33+....+....+1
1.2332.23332.33.23332.2332.1
1p3XX3.3XX3.33.3XX3.3XX3p1
1.2332.23332.22.23332.2332.1
1+....+..+..+..+..+..+....+1
1.2332.22.23333332.22.2332.1
1.2332.33.23399332.33.2332.1
1+....+33+..+33+..+33+....+1
011116.39332|33|23393.611110
XXXXX1.39332|22|23393.1XXXXX
XXXXX1.33n--n--n--n33.1XXXXX
XXXXX1.33|455==554|33.1XXXXX
111116.22|5XXXXXX5|22.611111
n-----+---n5XXXXXX5n--+-----n
111116.22|5XXXXXX5|22.611111
XXXXX1.33|45555554|33.1XXXXX
XXXXX1.33n-----n33.1XXXXX
XXXXX1.33|23333332|33.1XXXXX
011116.22|23399332|22.611110
1+....+..+..+33+..+..+....+1
1.2332.23332.33.23332.2332.1
1.2393.23332.22.23332.3932.1
1P.+33+..+..+--+..+..+33+.P1
832.33.22.23333332.22.33.237
732.22.33.23399332.33.22.238
1+..+..+33+..+33+..+33+..+..+1
1.2333399332.33.2339933332.1
1.2333333332.22.2333333332.1
1+.....+..+.....+1
0111111111111111111111111110
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Maze_rotation1.txt

```
.....
.....
.....
0000000000000000000000000003
1.....13.....3
```

```

1.0003.00003.13.00003.0003.3
1.1..3.1...3.13.1...3.1..3.3
1.1222.12222.12.12222.1222.3
1.....3
1.0003.03.00000003.03.0003.3
1.1222.13.12230222.13.1222.3
1.....13....13....13.....3
122223.11003.13.00023.022222
....1.10222.12.12233.3.....
....1.13.....13.3.....
....1.13.000..003.13.3.....
000002.12.1.....3.12.100000
.....1.....3.....
222223.03.1.....3.03.022222
....1.13.12222222.13.3.....
....1.13.....13.3.....
....1.13.00000003.13.3.....
000002.12.12230222.12.100003
1.....13.....3
1.0003.00003.13.00003.0003.3
1.1233.12222.12.12222.1022.3
1...13.....13...3
103.13.03.00000003.03.13.003
122.12.13.12230222.13.12.123
1.....13....13....13.....3
1.0000021003.13.0002100003.3
1.1222222222.12.1222222222.3
1.....3
122222222222222222222222222222
.....
.....

```