

Life Icons

The life icons will now show in the bottom-left part of the screen.

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

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.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.livesprites.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.texrgroup.showText(READYTXT)

def render(self):
    self.screen.blit(self.background, (0,0))
    self.nodes.render(self.screen)
    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()
```

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)

```