

End of Level

At this point, when all the pellets are eaten, the game will reset and bring them back. There is also a brief pause when Pacman eats a ghost.

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

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

    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))

    def update(self):
```

```

dt = self.clock.tick(30) / 1000.0
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 checkEvents(self):
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        elif event.type == KEYDOWN:
            if event.key == K_SPACE:
                self.pause.setPause(playerPaused=True)
            if not self.pause.paused:
                self.showEntities()
            else:
                self.hideEntities()

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.pause.setPause(pauseTime=1, func=self.showEntities)
                ghost.startSpawn()

def checkPelletEvents(self):
    pellet = self.pacman.eatPellets(self.pellets.pelletList)
    if pellet:
        self.pellets.numEaten += 1
        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.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.startGame()

    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)
        pygame.display.update()

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

```