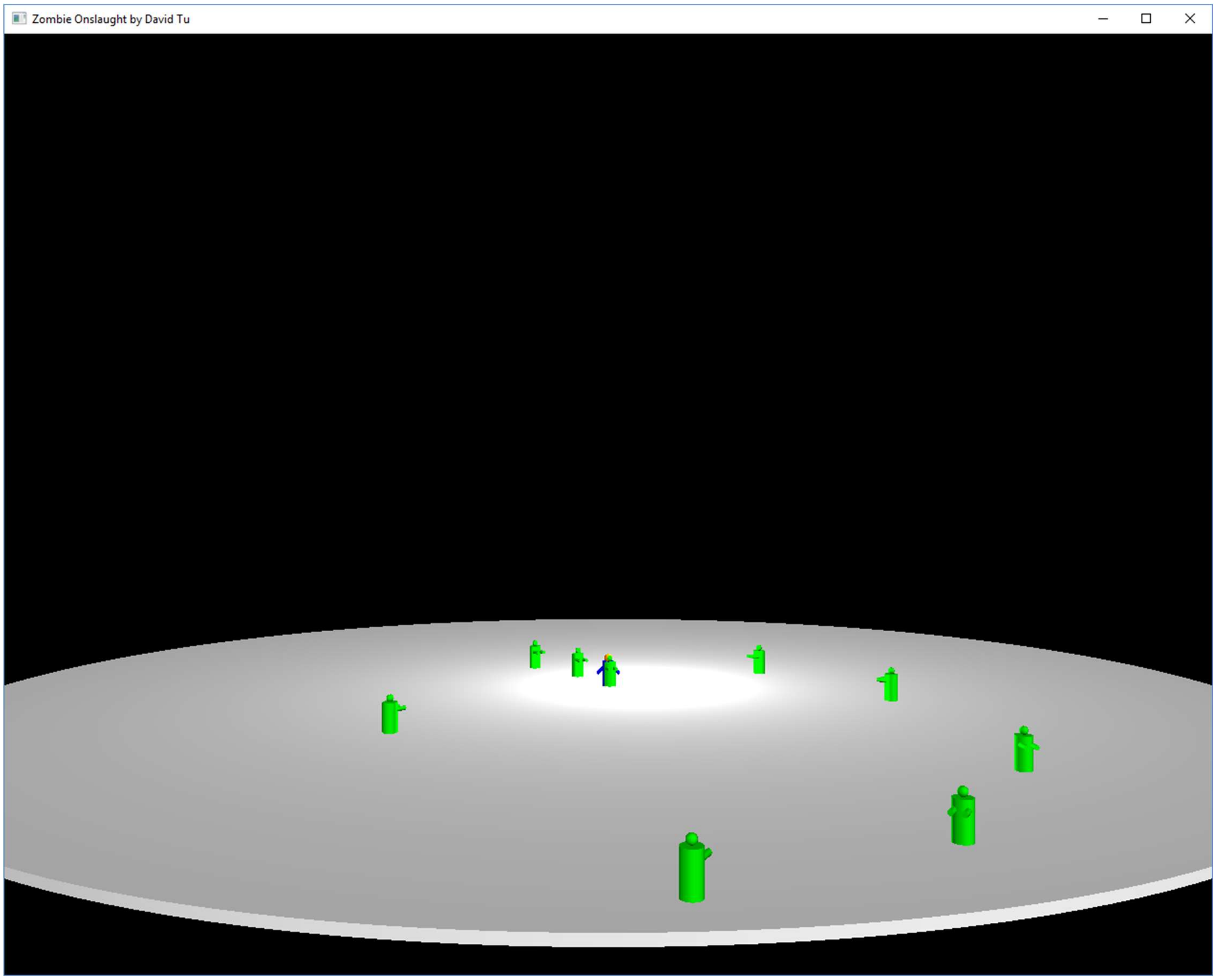
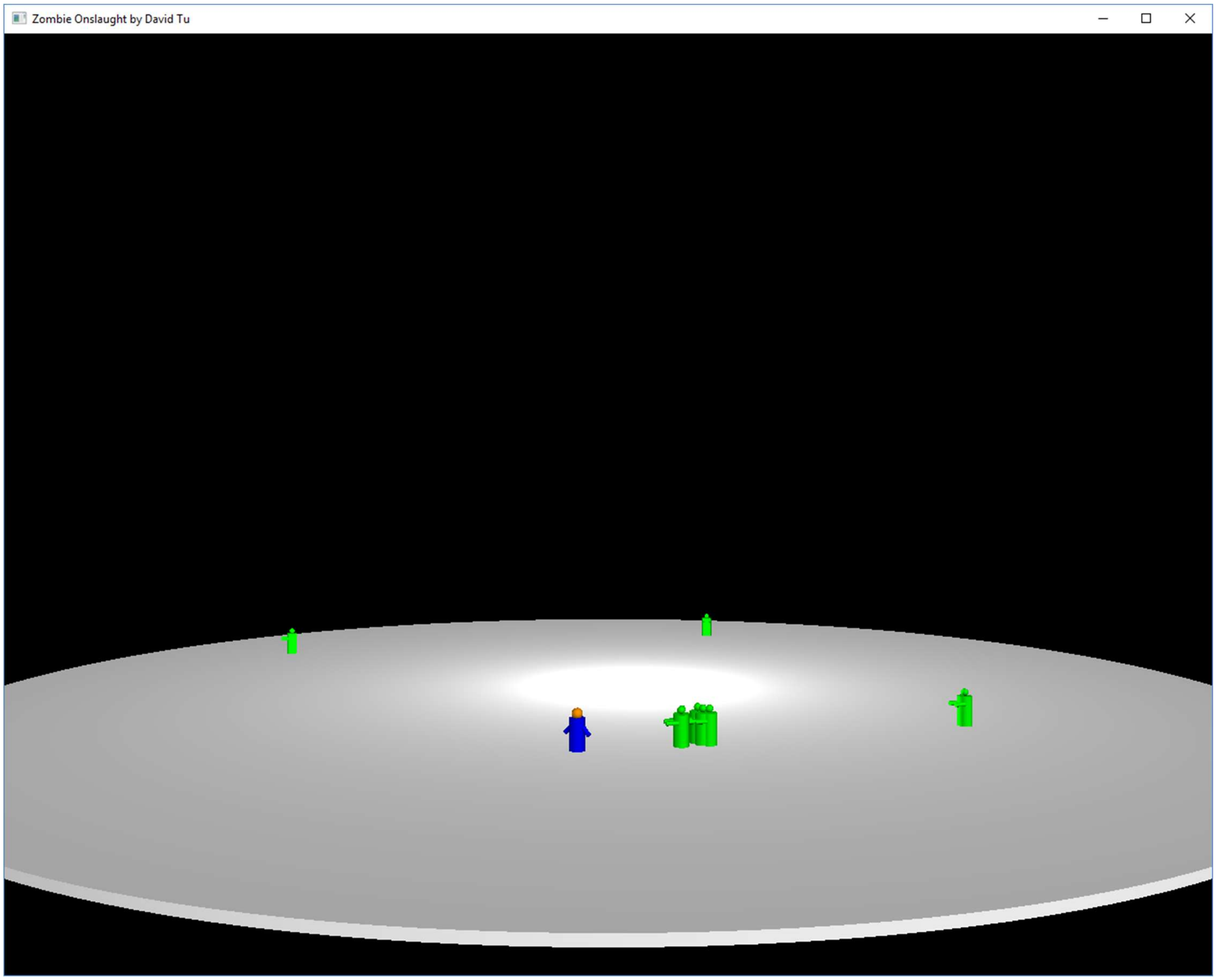
I. Source Code

```
"""
Zombie Onslaught v1.1.py

David Tu     david.tu2@csu.fullerton.edu

Description: A simple implementation of a 3rd person action game

Rules of the Game:
1. You attack by setting traps
2. If some enemy runs over a trap, they will be defeated
3. Defeat all the enemies to clear the game

How to Play:
SPACEBAR: Set a trap
LEFT ARROW: While held, the avatar will rotate left
RIGHT ARROW: While held, the avatar will rotate right
UP ARROW: While held, the avatar will move forward
DOWN ARROW: While held, the avatar will move backwards
C KEY: Toggles the camera between 1st and 3rd person view

Known Bugs:
1. The player can lay a trap on the same spot as an existing one
2. Sometimes, the player or a zombie can stay stuck on the bounds of the circle
"""
from visual import *
import math
import random

scene = display(title = "Zombie Onslaught by David Tu", width = 1280, height = 1024)
GROUND_RADIUS = 100
ZOMBIES = 5
STALKERS = 5

class Person:
    def __init__(self, position = vector(0, 0, 0), heading = vector(0, 0, 1), speed = 1):
        self.location = position
        self.heading = heading.norm() # make the length of the vector a unit vector, since this is a direction I am going
        self.speed = speed
        self.body_frame = frame(pos = self.location)
        self.body = cylinder(frame = self.body_frame, axis = (0, 4, 0), radius = 1) # all of those body parts are
        #positioned relative to the Person's position
        self.leftArm = cylinder(frame = self.body_frame, pos = vector(0.6, 3, 0), axis = (0, 0, 2), radius = .3)
        self.rightArm = cylinder(frame = self.body_frame, pos = vector(-0.6, 3, 0), axis = (0, 0, 2), radius = .3)
        self.head = sphere(frame = self.body_frame, pos = vector(0, 4.5, 0), radius = 0.5)

    def turn(self, angle): # changes the person's facing direction
        theta = math.radians(angle) # VPython assumes all angles are in radians
        self.heading = rotate(self.heading, angle = theta, axis = vector(0, 1, 0))
        self.body_frame.rotate(angle = theta, axis = vector(0, 1, 0), origin = self.location)

    def forward(self): # scale by the person's speed
        self.location += self.heading * self.speed
        self.body_frame.pos += self.heading * self.speed

    def backward(self):
        self.location -= self.heading * self.speed
        self.body_frame.pos -= self.heading * self.speed

class Zombie(Person): # inheritance
    def __init__(self, position = vector(0, 0, 0), heading = vector(0, 0, 1)):
        Person.__init__(self, position, heading) # call to superclass' construct
        self.body.color = color.green
        self.leftArm.color = color.green
        self.rightArm.color = color.green
        self.head.color = color.green

    def update(self): # an example of rotation then translation!
```

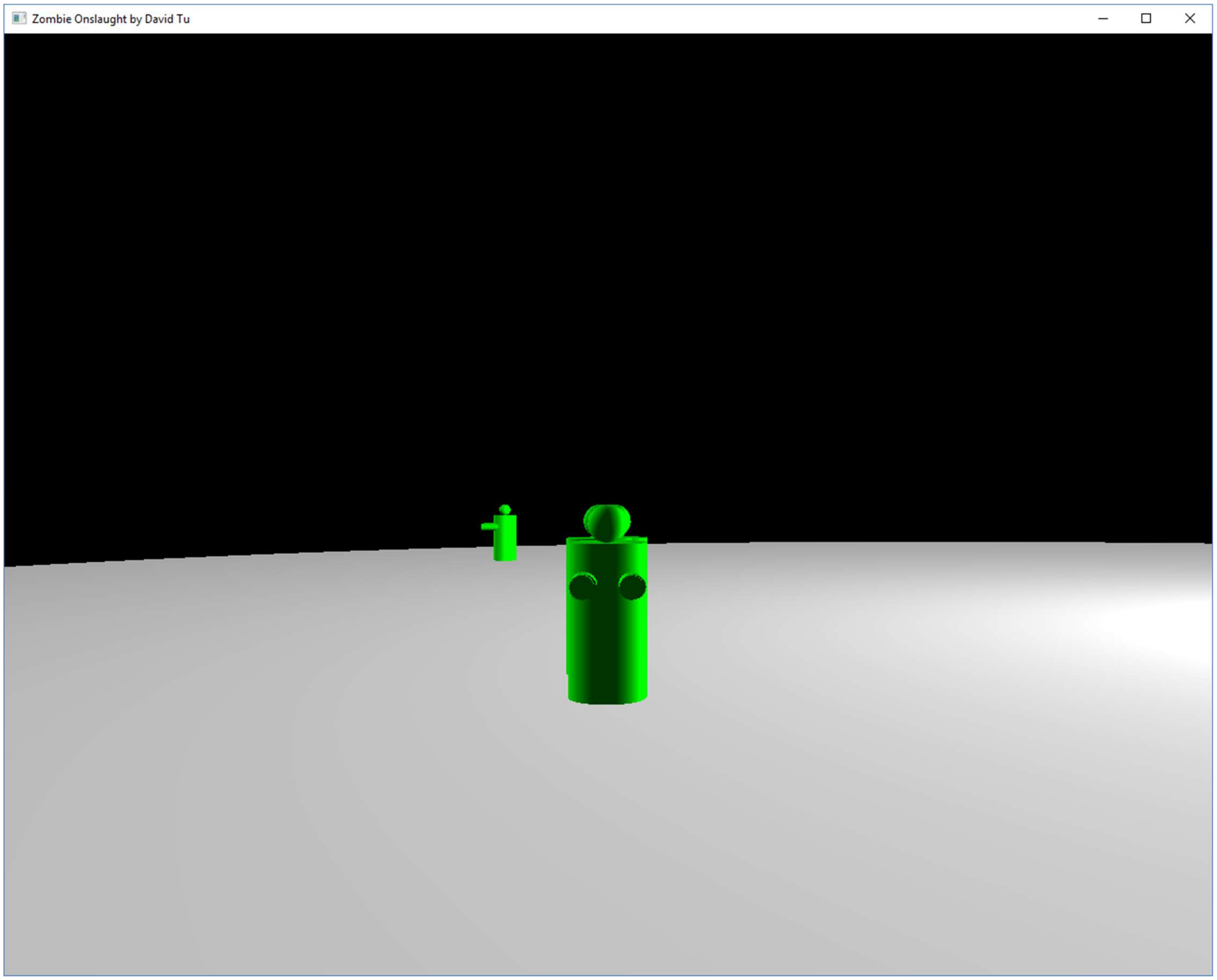
```
        self.turn(random.uniform(-5, 5)) # method that calls the turn method with a random angle between -5 and 5 degrees
        self.forward()

class Stalker(Person):
    def __init__(self, position = vector(0, 0, 0), heading = vector(0, 0, 1)):
        Person.__init__(self, position, heading)
        self.speed = .2
        self.body.color = color.green
        self.leftArm.color = color.green
        self.rightArm.color = color.green
        self.head.color = color.green

    def update(self, location):
        distance = location - self.location #distance to the player
        cosB = dot(distance, self.heading)/(mag(distance) * mag(self.heading))
        if cosB < -1 or cosB > 1: # workaround: arccos is only valid in [-1, 1]
            beta = acos(1)
        else:
            beta = acos(cosB) # returns radians
        angle = math.degrees(beta)
        self.turn(random.uniform(-angle, angle)) # makes the stalker turn either clockwise or counter clockwise
        self.forward()

class Hero(Person):
    def __init__(self, position = vector(0, 0, 0), heading = vector(0, 0, 1)):
        Person.__init__(self, position, heading)
        self.body.color = color.blue
        self.leftArm.pos = vector(0.8, 3, 0)
        self.leftArm.axis = (1, -1, 0)
        self.leftArm.color = color.blue
        self.rightArm.pos = vector(-0.8, 3, 0)
        self.rightArm.axis = (-1, -1, 0)
        self.rightArm.color = color.blue
        self.head.radius = 0.7
        self.head.color = color.orange
        self.camera = box(pos = self.location + vector(0, 5, 2), color = color.blue, length = 1.5, height = 1, width = 1,
opacity = False)

    def turn(self, angle):
        theta = math.radians(angle)
        self.heading = rotate(self.heading, angle = theta, axis = vector(0, 1, 0))
        self.body_frame.rotate(angle = theta, axis = vector(0, 1, 0), origin = self.location)
        self.camera.rotate(angle = theta, axis = vector(0, 1, 0), origin = self.location)

    def forward(self):
        self.location += self.heading * self.speed
        self.body_frame.pos += self.heading * self.speed
        self.camera.pos += self.heading * self.speed

    def backward(self):
        self.location -= self.heading * self.speed
        self.body_frame.pos -= self.heading * self.speed
        self.camera.pos -= self.heading * self.speed

def makeZombies():
    zombies = []
    for z in range(ZOMBIES):
        theta = random.uniform(0, 360)
        r = random.uniform(0, GROUND_RADIUS)
        x = r * cos(math.radians(theta))
        z = r * sin(math.radians(theta))
        zombies.append(Zombie(position = vector(x, 0, z)))
    return zombies

def makeStalkers():
    stalkers = []
    for s in range (STALKERS):
        theta = random.uniform(0, 360)
        r = random.uniform(0, GROUND_RADIUS)
        x = r * cos(math.radians(theta))
        z = r * sin(math.radians(theta))
        stalkers.append(Stalker(position = vector(x, 0, z)))
    return stalkers

def collision(object1, targetVector): #determines whether a point is within a sphere or not
    distance = mag(object1.pos - targetVector)
    if(distance < object1.radius):
        return True # means I've hit the bomb
    else:
        return False

def main():
    ground = cylinder(pos = (0, -1, 0), axis = (0, 1, 0), radius = GROUND_RADIUS)
    lightSource = local_light(pos = (5, 10, 0), color = color.white)
    player = Hero()
    zombies = makeZombies() # zombie container
    stalkers = makeStalkers() # stalker container
    traps = [] # traps container
    scene.autocenter = False
    scene.autoscale = False
    firstPersonView = False

    while True: # the game loop
        rate(30)
        if firstPersonView:
            scene.forward = player.heading
            scene.center = player.camera.pos
            scene.range = 1
        else:
            scene.center = vector(0, 30, 0)
            scene.fov = pi/3
            scene.range = 100

        # check for player input
        if scene.kb.keys: # is there an event waiting to be processed?
            key = scene.kb.getkey() # get keyboard information
            if key == 'left':
                player.turn(5)
            elif key == 'right':
                player.turn(-5)
            elif key == 'up':
                player.forward()
            elif key == 'down':
                player.backward()
            elif key == ' ': #lay a trap
                trap = sphere(pos = player.location, radius = 1, color = color.orange, material = materials.emissive)
                traps.append(trap)
                player.speed = 4 # will make the player step forward as they lay their trap
                player.forward()
                player.speed = 1
            elif key == 'c':
                if firstPersonView == True:
                    firstPersonView = False
                else:
                    firstPersonView = True

        if mag(player.location) >= GROUND_RADIUS:
            player.turn(180)
        for zombie in zombies:
            zombie.update()
            if mag(zombie.location) >= GROUND_RADIUS:
                zombie.turn(random.uniform(150, 210))
        for stalker in stalkers:
            stalker.update(player.location)
            if mag(stalker.location) >= GROUND_RADIUS:
                stalker.turn(random.uniform(150, 210))

        #check for collisions
        for trap in traps:
            for zombie in zombies:
                if collision(trap, zombie.location) and trap.visible == True:
                    zombie.body_frame.visible = False
                    trap.visible = False
                    zombies.remove(zombie)
                    del zombie
                    traps.remove(trap)
            for stalker in stalkers:
                if collision(trap, stalker.location) and trap.visible == True:
                    stalker.body_frame.visible = False
                    trap.visible = False
                    stalkers.remove(stalker)
                    del stalker
                    traps.remove(trap)
            del trap

if __name__ == '__main__': main()
```

II. Output

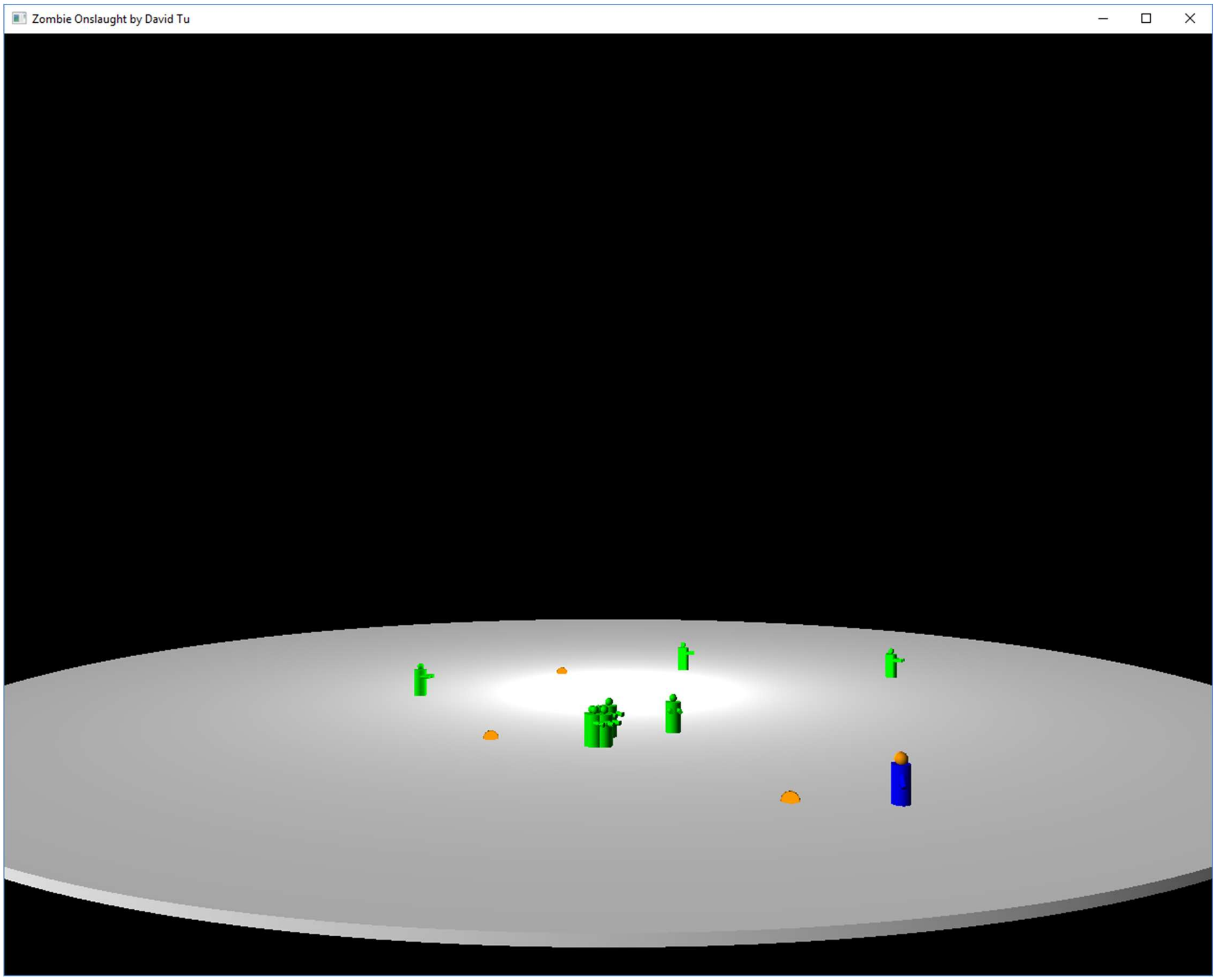




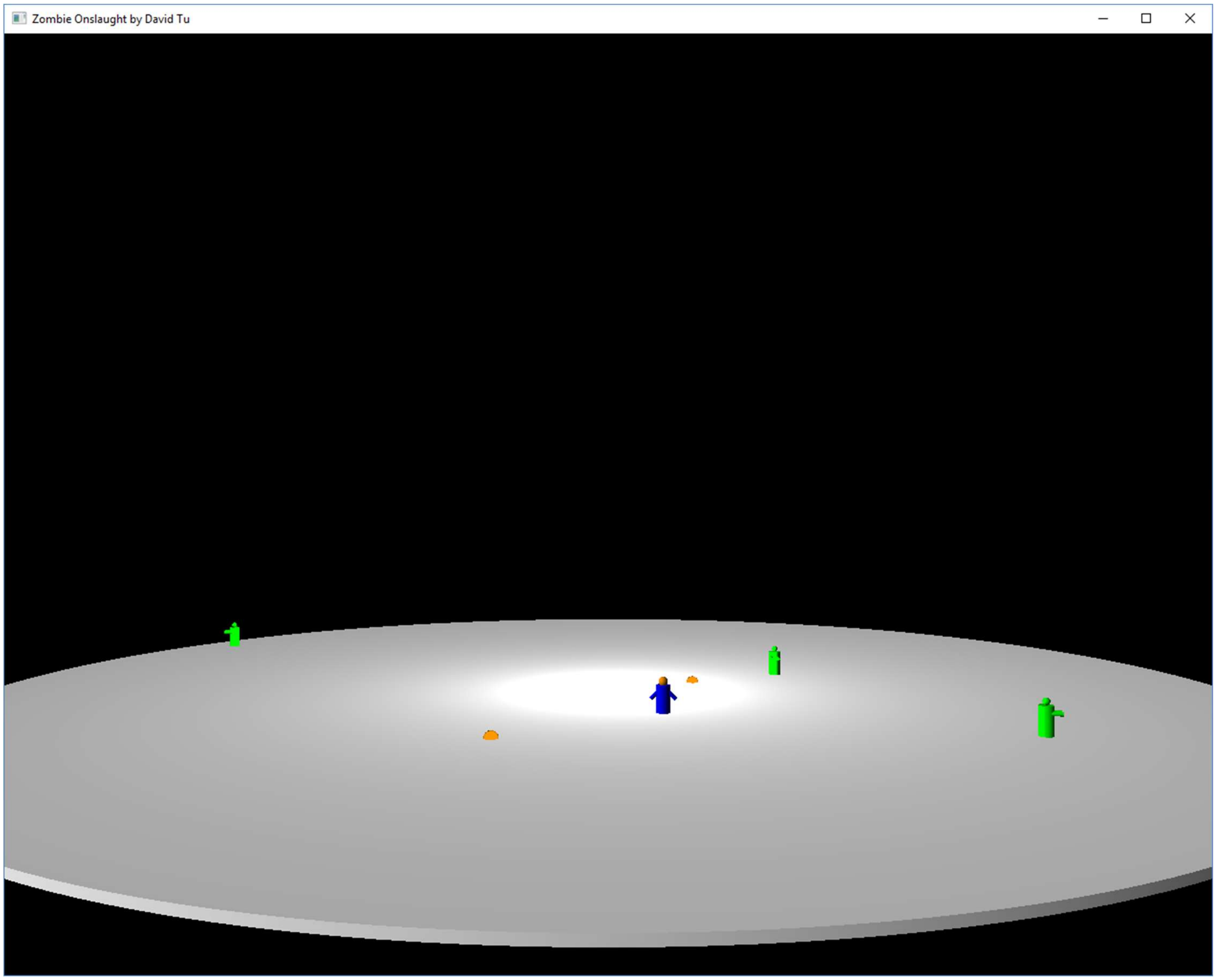
A zombie horde is following the player



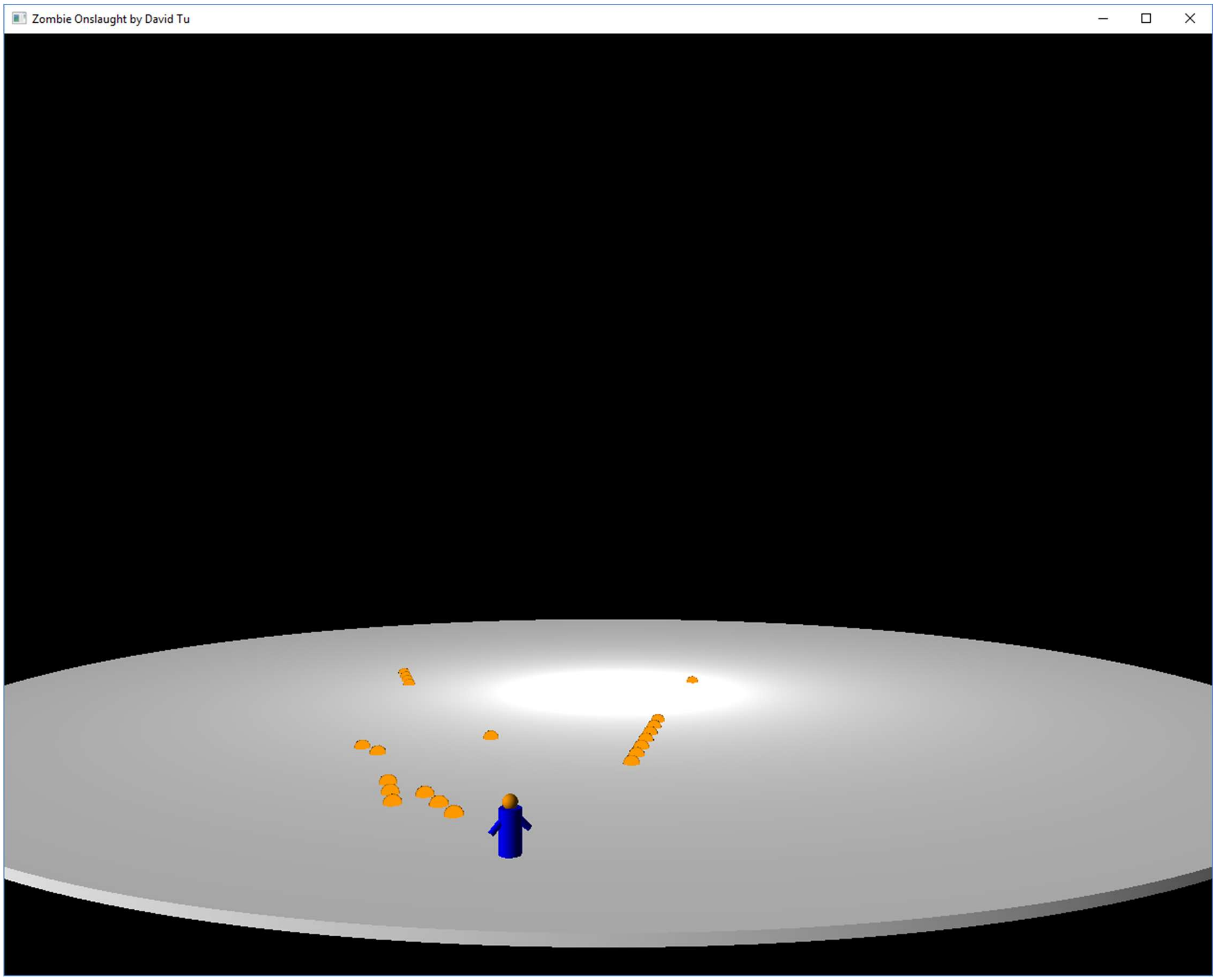
First Person view



Placing traps



Traps defeat the enemies



All the enemies are defeated