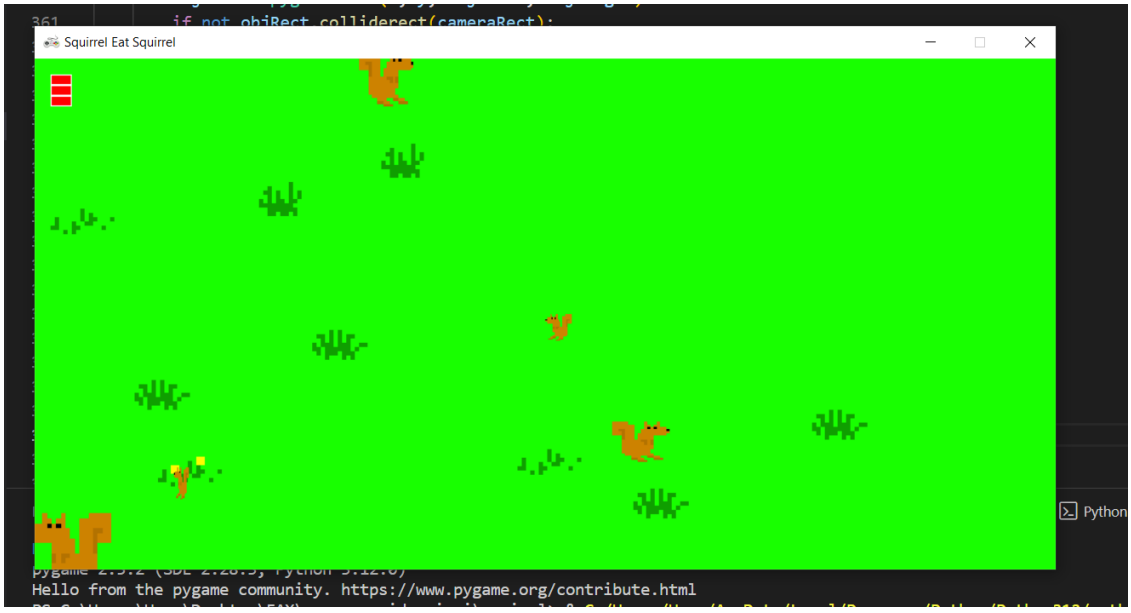
- Increase the camera area of the game to 960x480 pixels

```
WINWIDTH = 960 #1st req: Increase the camera area of the game
WINHEIGHT = 480 # height in pixels
HALF_WINWIDTH = int(WINWIDTH / 2)
```

- From this size

- To this



- Define two new constants for the CAMERASLACK constants for the horizontal and vertical direction appropriately

```
CSHORIZONTAL = 80   #2nd req: horizontal dir
CSVERTICAL = 60   #2nd req: vertical dir
MOVERATE = 9          # how fast the player moves
```

```
#2nd req: change CAMERASLACK with CSHORIZONTAL and CSVERTICAL
if (camerax + HALF_WINWIDTH) - playerCenterx > CSHORIZONTAL:
    camerax = playerCenterx + CSHORIZONTAL - HALF_WINWIDTH
elif playerCenterx - (camerax + HALF_WINWIDTH) > CSHORIZONTAL:
    camerax = playerCenterx - CSHORIZONTAL - HALF_WINWIDTH
if (cameray + HALF_WINHEIGHT) - playerCentery > CSVERTICAL:
    cameray = playerCentery + CSVERTICAL - HALF_WINHEIGHT
elif playerCentery - (cameray + HALF_WINHEIGHT) > CSVERTICAL:
    cameray = playerCentery - CSVERTICAL - HALF_WINHEIGHT
```

- Introduce the possibility an enemy squirrels to bounce downwards (the squirrel could bounce only in one direction only upwards or only downwards). The direction is determined when creating an enemy squirrel

```
# move all the squirrels
for sObj in squirrelObjs:
    # move the squirrel, and adjust for their bounce
    #3rd req:
    if sObj['randomSq'] == 1:
        sObj['y'] += sObj['movey']
    else:
        sObj['x'] += sObj['movex']
        sObj['y'] += sObj['movey']
        sObj['bounce'] += 1
    if sObj['bounce'] > sObj['bouncerate']:
        sObj['bounce'] = 0 # reset bounce amount
```

- Marking the squirrelsfrom 1 to 5 randomly, when number 2 squirrel is generated the movement is only vertical

```
def makeNewSquirrel(camerax, cameray):
    sq = {}
    generalSize = random.randint(5, 25)
    multiplier = random.randint(1, 3)
    sq['width']  = (generalSize + random.randint(0, 10)) * multiplier
    sq['height'] = (generalSize + random.randint(0, 10)) * multiplier
    sq['x'], sq['y'] = getRandomOffCameraPos(camerax, cameray, sq['width'], sq['height'])
    sq['movex'] = getRandomVelocity()
    sq['movey'] = getRandomVelocity()
    sq['randomSq'] = random.randint(1, 5)  #3rd req
    if sq['movex'] < 0: # squirrel is facing left
        sq['surface'] = pygame.transform.scale(L_SQUIR_IMG, (sq['width'], sq['height']))
    else: # squirrel is facing right
        sq['surface'] = pygame.transform.scale(R_SQUIR_IMG, (sq['width'], sq['height']))
    sq['bounce'] = 0
    sq['bouncerate'] = random.randint(10, 18)
    sq['bounceheight'] = random.randint(10, 50)
    return sq
```

- How big the player need to be to lose

```
WINSIZE = 300          # how big the player needs to be to win
LOSTSIZE = 5           #4th req: how big the player needs to be to lose
INVULNTIME = 2         # how long the player is invulnerable after being hit in s
```

```
    elif not invulnerableMode:
        # player is smaller and takes damage
        invulnerableMode = True
        invulnerableStartTime = time.time()
        #4th req: decreasing the size of the squirrel
        playerObj['size'] -= int((sqObj['width'] * sqObj['height']) ** 0.2) + 1
        #playerObj['health'] -= 1
        if playerObj['size'] <= LOSTSIZE:
            gameOverMode = True # turn on "game over mode"
            gameOverStartTime = time.time()

me is over, show "game over" text
LAYSURF.blit(gameOverSurf, gameOverRect)
ime.time() - gameOverStartTime > GAMEOVERTIME:
return # end the current game
```