

# Squirrel Game

## Frame

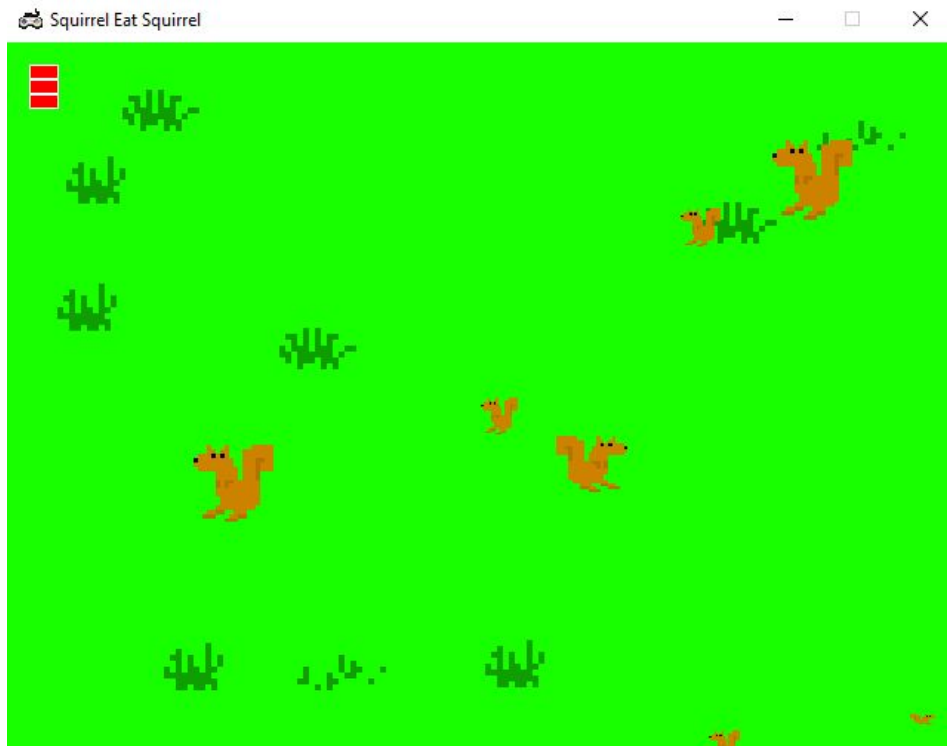
- Icon, title, quit

## Objects

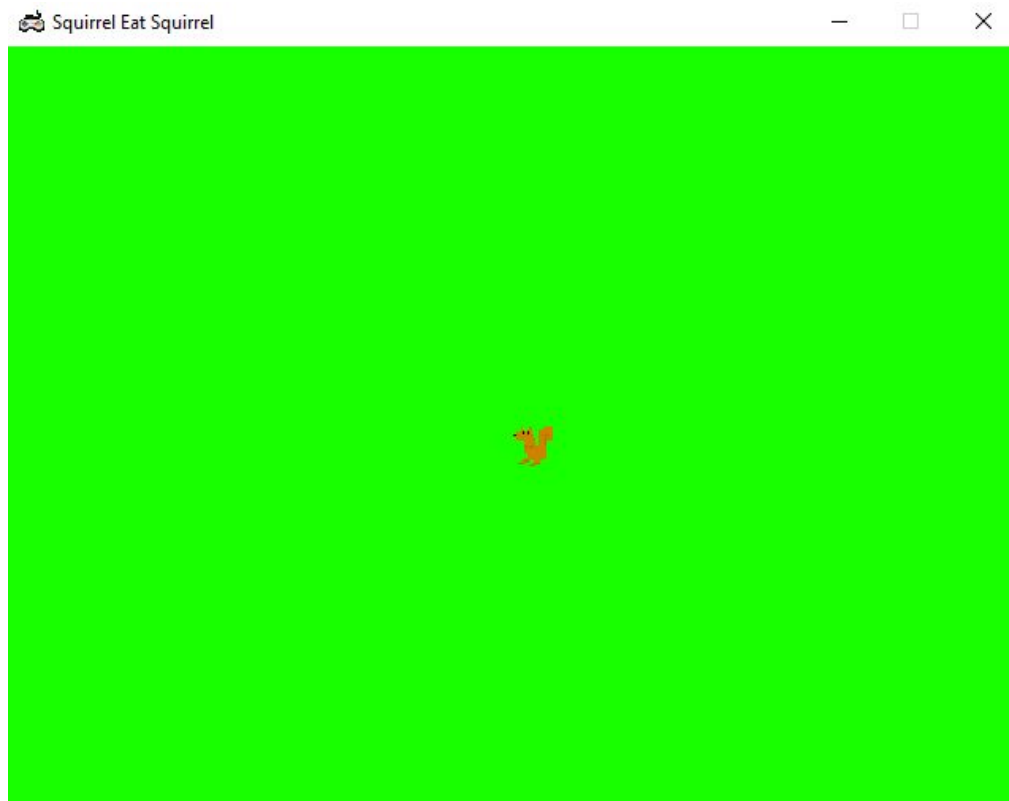
- Player squirrel
- Enemy squirrel
- Grass

## Rules

- Eat smaller squirrel to grow
- Hit bigger squirrel to lose health
- Become Omega squirrel to win
- Arrow keys or AWSAD keys,
  - Key down to move
  - Key up to stop



# Project 1:



# Code

```
FPS = 30 # frames per second to update the screen
WINWIDTH = 640 # width of the program's window, in pixels
WINHEIGHT = 480 # height in pixels
HALF_WINWIDTH = int(WINWIDTH / 2)
HALF_WINHEIGHT = int(WINHEIGHT / 2)

GRASSCOLOR = (24, 255, 0)

MOVERATE = 9           # how fast the player moves
BOUNCERATE = 6         # how fast the player bounces (large is slower)
BOUNCEHEIGHT = 30      # how high the player bounces
STARTSIZE = 25         # how big the player starts off
LEFT = 'left'
RIGHT = 'right'
```

```
def main():
    global FPSCLOCK, DISPLAYSURF, L_SQUIR_IMG, R_SQUIR_IMG

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    pygame.display.set_icon(pygame.image.load('gameicon.png'))
    DISPLAYSURF = pygame.display.set_mode((WINWIDTH, WINHEIGHT))
    pygame.display.set_caption('Squirrel Eat Squirrel')

    # load the image files
    L_SQUIR_IMG = pygame.image.load('squirrel.png')
    R_SQUIR_IMG = pygame.transform.flip(L_SQUIR_IMG, True, False)

    while True:
        runGame()
```

```
def runGame():  
  
    # stores the player object:  
    playerObj = {'surface': pygame.transform.scale(L_SQUIR_IMG, (STARTSIZE, STARTSIZE)),  
                 'facing': LEFT,  
                 'width': STARTSIZE,  
                 'height': STARTSIZE,  
                 'x': HALF_WINWIDTH,  
                 'y': HALF_WINHEIGHT,  
                 'bounce': 0,  
                 'bouncerate': BOUNCERATE,  
                 'bounceheight': BOUNCEHEIGHT}  
  
    moveLeft = False  
    moveRight = False  
    moveUp = False  
    moveDown = False
```

```
while True: # main game loop
    DISPLAYSURF.fill(GRASSCOLOR)
    moveLeft, moveRight, moveUp, moveDown = eventProcess(moveLeft, moveRight, moveUp, moveDown)
    if moveLeft or moveRight or moveUp or moveDown:
        if moveLeft:
            playerObj['x'] -= MOVERATE
            if playerObj['facing'] != LEFT:
                playerObj['surface'] = pygame.transform.scale(L_SQUIR_IMG, (playerObj['width'], playerObj['height']))
                playerObj['facing'] = LEFT
        if moveRight:
            playerObj['x'] += MOVERATE
            if playerObj['facing'] != RIGHT:
                playerObj['surface'] = pygame.transform.scale(R_SQUIR_IMG, (playerObj['width'], playerObj['height']))
                playerObj['facing'] = RIGHT
        if moveUp:
            playerObj['y'] -= MOVERATE
        if moveDown:
            playerObj['y'] += MOVERATE

    if (moveLeft or moveRight or moveUp or moveDown) or playerObj['bounce'] != 0:
        increaseBounce(playerObj)

    displaySquirrel(playerObj)
    pygame.display.update()
    FPSCLOCK.tick(FPS)
```

```
def eventProcess(moveLeft, moveRight, moveUp, moveDown):  
    for event in pygame.event.get(): # event handling loop  
        if event.type == QUIT:  
            terminate()  
        elif event.type == KEYDOWN:  
            if event.key == K_UP:  
                moveUp = True  
            elif event.key == K_DOWN:  
                moveDown = True  
            elif event.key == K_LEFT:  
                moveLeft = True  
            elif event.key == K_RIGHT:  
                moveRight = True  
        elif event.type == KEYUP:  
            if event.key == K_LEFT:  
                moveLeft = False  
            elif event.key == K_RIGHT:  
                moveRight = False  
            elif event.key == K_UP:  
                moveUp = False  
            elif event.key == K_DOWN:  
                moveDown = False  
    return moveLeft, moveRight, moveUp, moveDown
```



```
def terminate():
    pygame.quit()
    sys.exit()

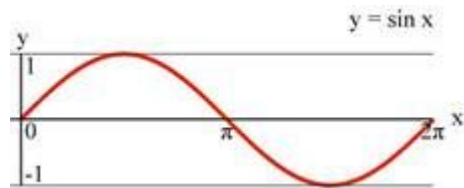
def increaseBounce(sObj):
    sObj['bounce'] += 1
    if sObj['bounce'] > sObj['bouncerate']:
        sObj['bounce'] = 0 # reset bounce amount

def displaySquirrel(sObj):
    sObj['rect'] = pygame.Rect(
        (sObj['x'], sObj['y'] - getBounceAmount(sObj['bounce'], sObj['bouncerate'], sObj['bounceheight']),
        sObj['width'], sObj['height']))
    DISPLAYSURF.blit(sObj['surface'], sObj['rect'])

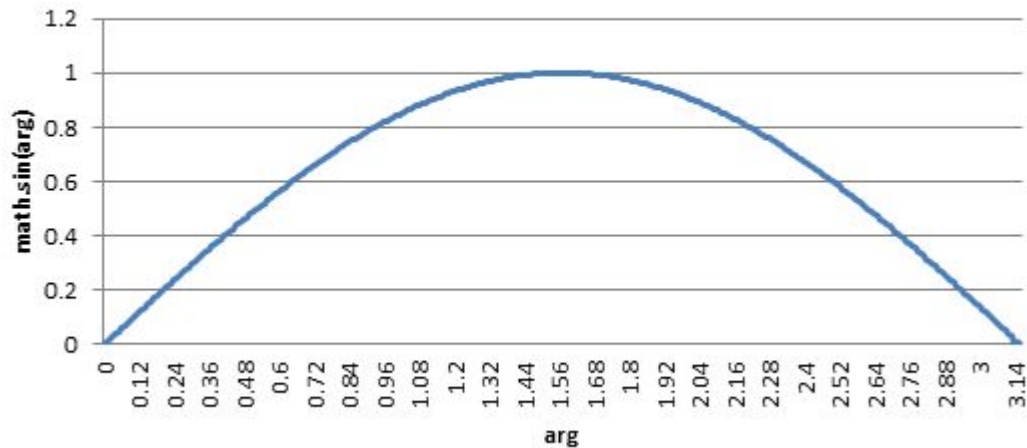
def getBounceAmount(currentBounce, bounceRate, bounceHeight):
    return int(math.sin( (math.pi / float(bounceRate)) * currentBounce ) * bounceHeight)

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

```
def getBounceAmount(currentBounce, bounceRate, bounceHeight):  
    return int(math.sin( (math.pi / float(bounceRate)) * currentBounce ) * bounceHeight)
```

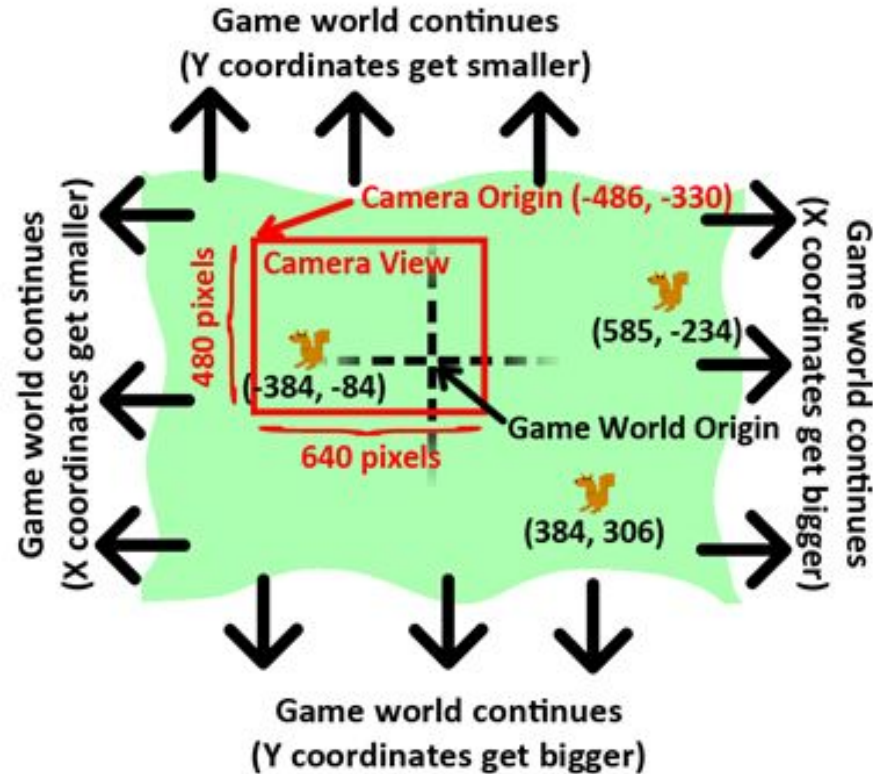


Values Returned by math.sin()



## Project 2:

# Game World and Camera View



# Active Area

<http://invpy.com/mariocamera>

