Politécnico do Porto

Escola Superior de Media Artes e Design

Carlos Pinto Guedes & Elói João Martins Leitão

Another Perspective of Pacman

Licenciatura em Tecnologias e Sistemas de Informação Web

Física Aplicada à Programação

Orientação: Prof.(ª) Doutor(a) Campos Neves

Vila do Conde, Novembro de 2018

SUMÁRIO

Este trabalho tem como conceito criar um “jogo” onde apliquemos os conceitos que aprendemos durante as aulas de Física Aplicada à Programação.

Assim, definiu-se como tema principal deste jogo o “Another Perspective of Pacman” (“Outra Perspetiva do Pacman”).

Espera-se que no final do relatório consiga-se compreender o conceito e os métodos usados para a realização do projeto. que este trabalho seja bem-sucedido, onde tentamos ser o mais original possível.

Obrigado,

Carlos Guedes & Elói Leitão

INDÍCE

0 INTRODUÇÃO

Este trabalho trata-se de uma realidade invertida do “Pacman” em que o objetivo é que o jogador, através do tradicional fantasma, consiga comer o máximo de “Pacman’s” possíveis evitando que estes passem a linha limitadora e evitando comer as cerejas.

Este projeto pode ser divido em três princípios essenciais tais como: o lançamento do fantasma (em que é possível controlar a velocidade e a direção deste), a criação e desenho dos objetos do jogo (“Pacman’s”, cerejas, ajudas, vidas e o fantasma) e, por último, uma pequena explicitação sobre as colisões existentes.

1 OBJETOS DO JOGO

Este capítulo foca-se numa pequena apresentação dos objetos presentes na tela do jogo.

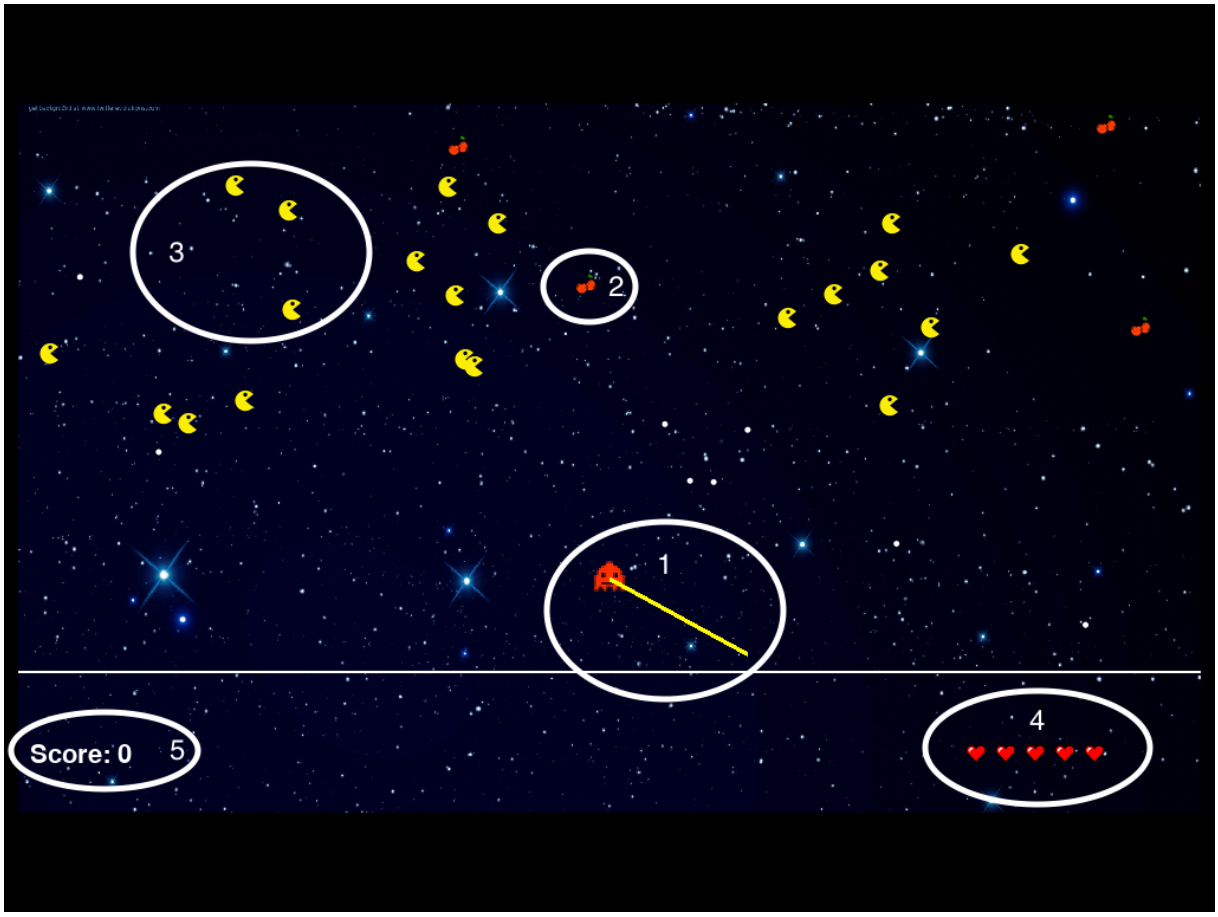

Figura 1 - ECRÃ DO JOGO

1.1 JOGADOR

O jogador é representado por um monstro (“IMG/player.png”), e pela linha que representa a velocidade que o jogador aplicará ao monstro.

1.2 CEREJA

A cereja (“IMG/cherrie.png”) é o que faz o jogador perder uma das vidas que obtém no início do jogo.

1.3 PACMAN

O Pacman é o nosso inimigo neste jogo (“IMG/inimigo.png”). Objetivo do jogar é comer o maior número de inimigos para ganhar pontos.

1.4 VIDAS

Aqui são representadas as nossas vidas iniciais. O jogador começa com cinco e o jogo acaba quando as vidas acabarem.

1.5 SCORE

O score é representado pelos pontos que o jogador obtém ao longo do jogo. A cada vez que o jogador “come” um inimigo recebe 1000 pontos, e quando comemos uma cereja perdemos 2000 pontos.

1.6 LIFE

As vidas que andarão juntas aos inimigos (“IMG/heart.png”) irão ajudar o jogador a prosseguir o jogo com maior facilidade, onde “comer” uma adicionará uma vida às vidas existentes (máximo de 5 vidas que o jogador pode ter).

2 CÓDIGO IMPLEMENTADO

Neste capítulo irá explicar-se minimamente (mas de maneira que se compreenda), o código implementado no projeto.

2.1 LANÇAMENTO DO JOGADOR

```
def getAceleration(x,y,x1,y1):

    d = math.sqrt(((x-x1)**2)+((y-y1)**2))

    return d

def getAngle(x,y,x1,y1):

    dy = y - y1

    dx = x - x1

    angulo = math.atan2(dy,dx)

    return angulo
```

Figura 2 - Definição das funções

Esta figura demonstra as funções que se usou para calcular a velocidade com que o jogador irá lançar o seu monstro.

2.1.1 getAceleration

Esta função recebe a posição do centro do círculo e a posição do rato, e retorna a distância entre os dois.

2.1.2 getAngle

Esta função retorna o ângulo entre a posição do rato e a posição do centro do círculo. Assim, sabemos o ângulo que o jogador irá tomar

2.1.3 Lançamento

```
v1 = int(distance * dt)
vx = v1
vy = v1


xposVetor, yposVetor = pygame.mouse.get_pos()

if event.type == pygame.MOUSEBUTTONDOWN:
    distance = getAceleration(xpos,ypos,xposVetor,yposVetor);
    ang = getAngle(xpos,ypos,xposVetor,yposVetor);
    cx = int(vx * math.cos(ang));
    yx = int(vy * math.sin(ang));
    label = 1
```

Figura 3 – Lançamento

Aqui, quando o evento do “click" do rato é acionado, calculamos a distância entre o ponto do rato e o centro do jogador. Aí, verifica-se o ângulo e calcula-se assim a incrementação que para o x e o y (isto é o que fará o jogador mover-se na tela).

2.2 DESENHO DOS OBJETOS

```
def drawLifes(vidas):
    x=comp_win*0.8
    for y in range(0,vidas):
        #pygame.draw.rect(win, (255,128,0), (x,alt_win*0.9,20,20))
        win.blit(lifeImg,(x,alt_win*0.9))
        x+=25
```

Figura 4 - DESENHO DAS VIDAS

Isto é uma criação estática de imagens, onde representa as vidas que o jogador ainda tem (que estão posicionadas no canto inferior direito).

A diferença entre estas vidas e os inimigos é o facto de os inimigos estarem sempre se a mover, enquanto que “estas” vidas estarão sempre paradas durante o jogo (o que muda é o número de vidas do jogador).

Figura 5 - CRIAÇÂO DOS INIMIGOS

```
#numero de inimigos
nInimigos=20

def drawInimigos(nInimigos):

    inimigos= [nInimigos]


    for x in range(-1, nInimigos-1):

        ini_x=random.randint(0,comp_win)

        ini_y=random.randint(0,alt_win*0.45)

        inimigos.append(pygame.Rect(ini_x, ini_y, raio, raio))

    return inimigos

inimigos_ecra=drawInimigos(nInimigos)
```

Aqui está representado um exemplo de como criamos a lista que contém os inimigos que irão aparecer na tela. Estes têm posições aleatórias.

Este método é também usado para criar as cerejas, e as vidas que o jogador “comer”.

```
#desenhar inimigos
for x in range(1, len(inimigos_ecra)):

    #quando chegarem ao limite do ecra passam para baixo
    if inimigos_ecra[x].left>=comp_win:inimigos_ecra[x].top+=40; inimigos_ecra[x].left=0
    #guardar coordenadas do inimigo
    coordX = inimigos_ecra[x].left
    coordY = inimigos_ecra[x].top
    coordX=coordX+speed
    #desenhar inimigo
    #pygame.draw.rect(win, (0,0,0), (coordX,coordY,raio,raio))
    win.blit(inImg,(coordX,coordY))
    inimigos_ecra[x].left=coordX
```

Figura 6 - DESENHO DOS INIMIGOS NA TELA

Com este pedaço de código, consegue-se mostrar o conteúdo da lista que contém os inimigos, onde decidiu-se, para uma primeira implementação (ver melhor capítulo 3), criar Retângulos, onde depois substituíamos por imagens, daí estar em comentário.

2.3 COLISÕES

Relativamente às colisões, baseiam-se em verificar sempre se o jogador ultrapassa os limites de cada inimigo, da tela e também do limite da área de jogo.

Para revisão do código, verificar em Anexos (Anexo 1).

3 PRIMEIRA IMPLEMENTAÇÃO

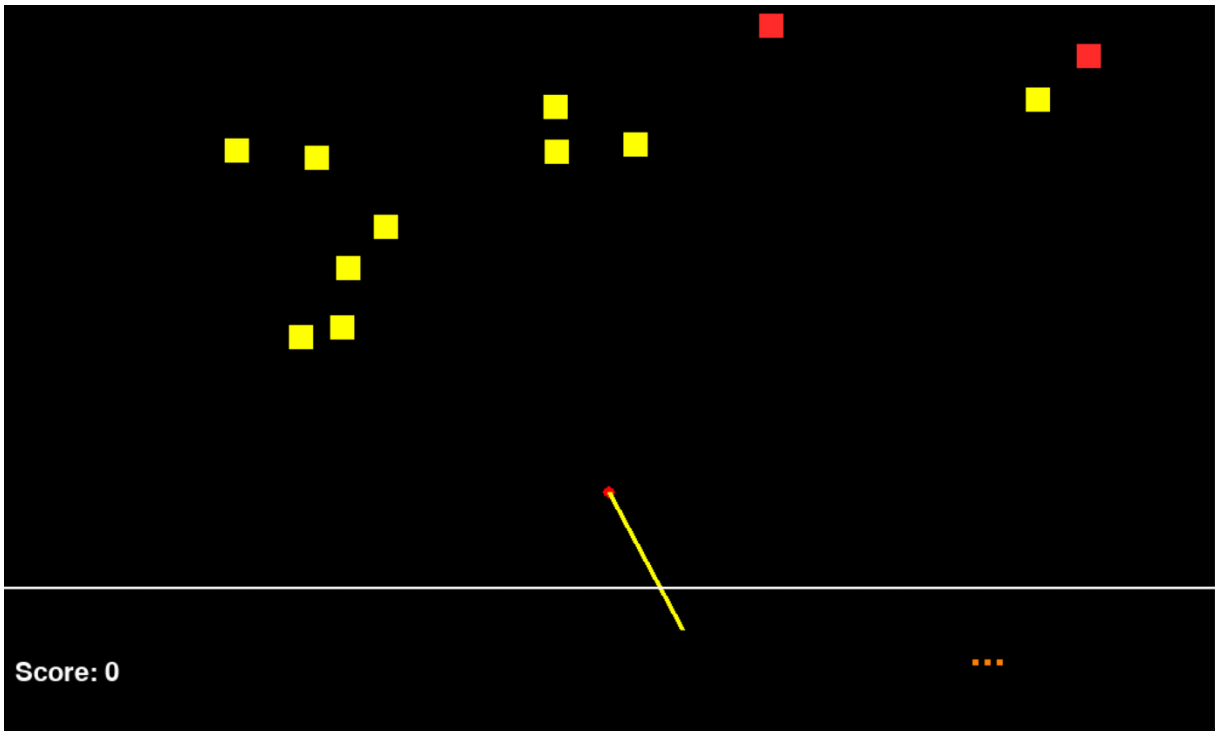

Figura 7 - PRIMEIRA IMPLEMENTAÇÃO

Na nossa primeira implementação, usamos Rects tem todos os objetos, pois assim seria mais fácil testarmos o código e implementarmos as primeiras ideias.

Depois de conseguirmos implementar todas as funcionalidades, passamos a inserir as imagens (para tornar o jogo mais atrativo).

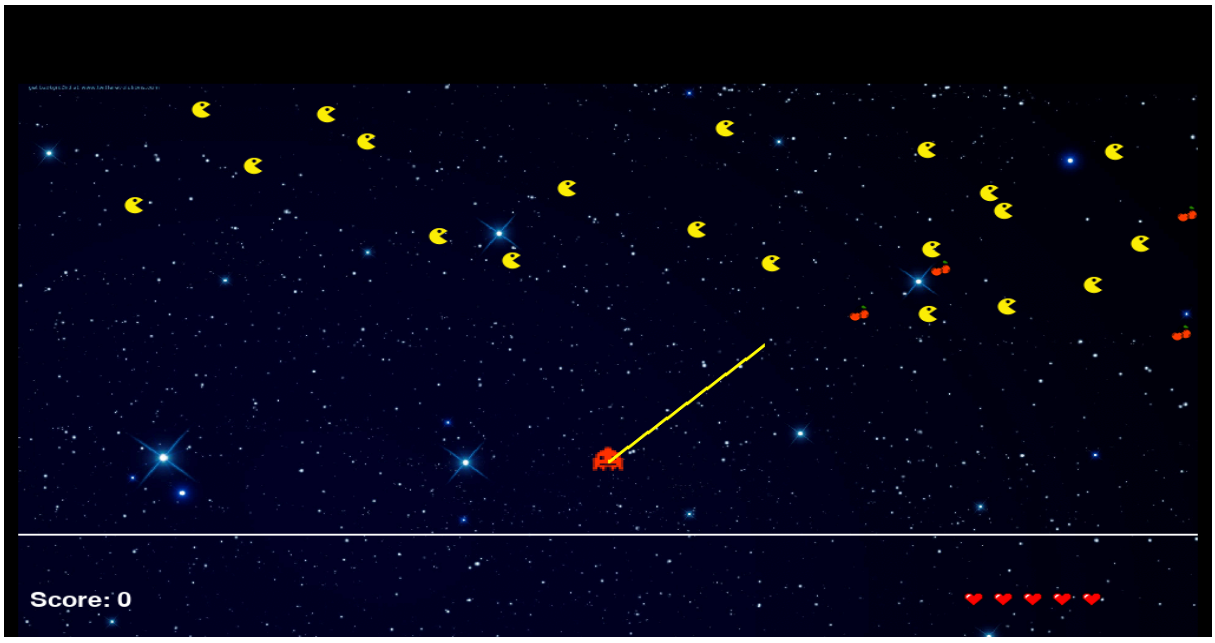

Figura 8 - IMPLEMENTAÇÃO FINAL

4 CONCLUSÃO

A realização deste trabalho permitiu o desenvolvimento e aplicação de conhecimentos relativamente as fórmulas físicas estudadas nas aulas, e também possibilitou um maior conhecimento da linguagem de programação “python”.

Este trabalho também “puxou” a nossa parte criativa, onde pensamos que o trabalho teve o resultado esperado.

5 REFERÊNCIAS BIBLIOGRÁFICAS

https://stackoverflow.com/questions/21209496/getting-width-and-height-of-an-image-in-pygame?rq=1 -> Altura de uma Imagem

https://www.pygame.org/docs/ -> Pequenas questões sobre código

6 ANEXOS

```
for x in range(1,len(inimigos_ecra)):


    coordIniX=inimigos_ecra[x].left
    coordIniY=inimigos_ecra[x].top

    if (xpos+raioJog >= (coordIniX ) and ypos + raioJog >= (coordIniY) and ypos - raioJog <= (coordIniY + raio) and xpos - raioJog <= (coordIniX+raio))and(label==1):

        del inimigos_ecra[x]
        raioJog+=1
        #playerImg=pygame.transform.scale(playerImg,(playerImg.get_width()+1,playerImg.get_height()+1))
        currentScale+=1
        scaled=pygame.transform.scale(playerImg,( currentScale,currentScale))
        new_rect=scaled.get_rect()
        win.blit(scaled,(xpos-new_rect.center[0],ypos-new_rect.center[1]))
        score+=1000
        #gerar posições random para mais um inimigo
        ini_x=random.randint(0,comp_win)
        ini_y=random.randint(0,alt_win*0.20)
        #adicionar ao array inimigos_ecra
        inimigos_ecra.append(pygame.Rect(ini_x, ini_y, raio, raio))
        break

    if  coordIniY+raio>=(alt_win*0.8):
        vidas-=1
        del inimigos_ecra[x]
        #gerar posições random para mais um inimigo
        ini_x=random.randint(0,comp_win)
        ini_y=random.randint(0,alt_win*0.20)
        #adicionar ao array inimigos_ecra
        inimigos_ecra.append(pygame.Rect(ini_x, ini_y, raio, raio))
        break
```

Anexo 1