Politécnico do Porto

Escola Superior de Media Artes e Design

Carlos Pinto Guedes & Elói João Martins Leitão

The Gear Game

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 “The Gear Game”.

Espera-se que no final do relatório o leitor consiga compreender os métodos utilizados no projeto.

Obrigado,

Carlos Guedes & Elói Leitão

INDÍCE

0 INTRODUÇÃO

A ideia para este projeto surgiu quando começamos a estudar VPython (GlowScript), onde aprendemos como construir uma roldana (gear) e, a partir daí, foi-nos surgindo ideias e o jogo final apareceu.

O objetivo é conseguir fazer com que as bolas passem a meta, sem tocar nas roldanas nem que toquem nos inimigos.

Aqui se encontra o link para o nosso jogo (http://www.glowscript.org/#/user/CarlosGuedes/folder/MyPrograms/program/TheGearGame/edit), onde o leitor poderá examinar o código, como também experimentá-lo.

1 OBJETOS DO JOGO

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

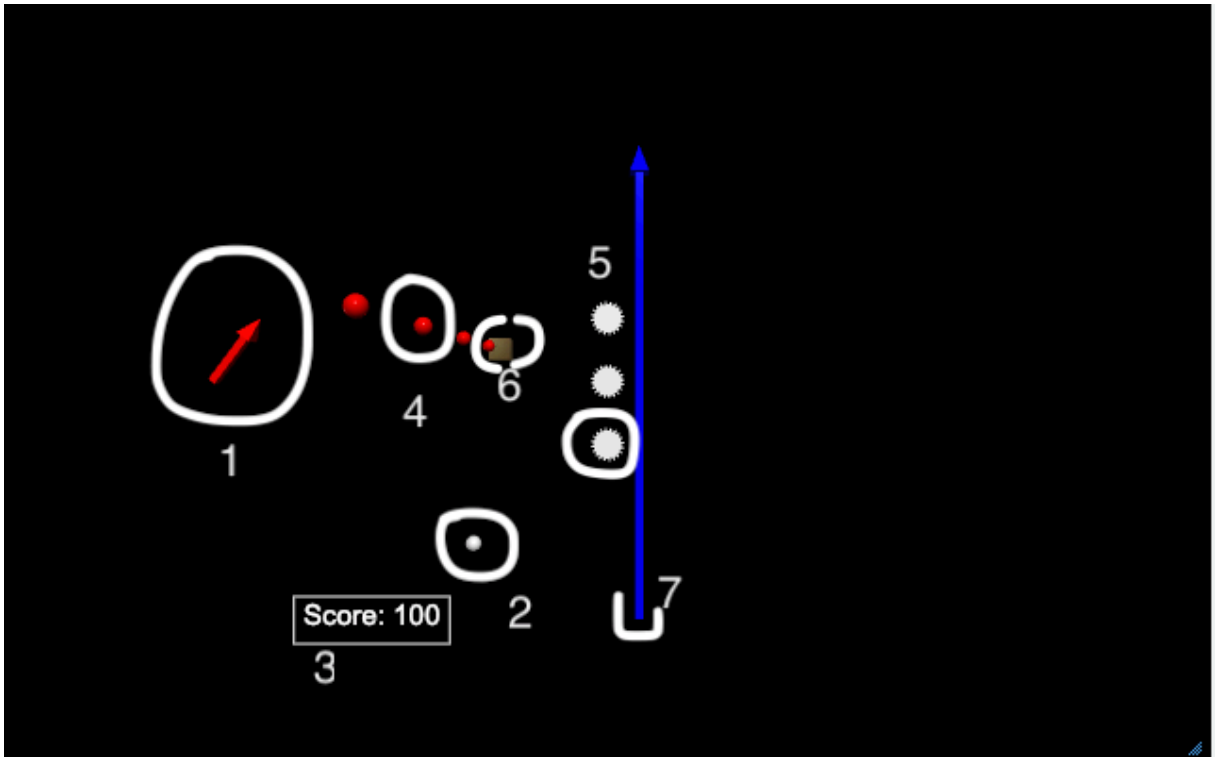

Figura 1 - ECRÃ DO JOGO

1.1 JOGADOR (ARROW)

O jogador é representado por “seta”, onde tem de clicar no butão “New Ball” para disparar uma bola (2).

1.2 BOLA

A bola é o elemento principal do jogo, onde ela tem de passar a meta (7) para podermos ganhar 100 pontos.

1.3 SCORE

O Score é uma label que nos informa do nossa pontuação atual.

1.4 INIMIGO

A cada 20 Frames é criado um inimigo a partir da “CreationBox”, onde se a bola do jogador coincidir com o inimigo, o jogador perder 500 pontos.

1.5 ROLDANA (GEAR)

A roldana também se inclui nos inimigos, mas estas estão estáticas (onde só rodam).

1.6 CREATION BOX

Esta caixa serve para os inimigos terem um sítio onde “nascer”. Por isso demos-lhe o nome de “Creation Box”.

1.7 META

Linha final onde as bolas lançadas pelo jogador têm que passar por ali, para ganhar 100 pontos.

2 CÓDIGO IMPLEMENTADO

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

2.1 LANÇAMENTO DA BOLA

```
def click():

    ball = sphere(pos=vec(-20,pointer[0].axis.y,0),size=vec(1,1,1))
    ball.velocity = vector(5,pointer[0].axis.y,0)

    balls.push(ball)
```

Figura 2 - Criação da Nova Bola

```
for k in range (balls.length): #Interseptions

        balls[k].pos = balls[k].pos + balls[k].velocity*dt
```

Figura 3 - Atualização da Posição das Bolas

A bola move-se em ordem a X, por isso aplicamos uma velocidade de X=5, e o Y fica com o valor que vem da seta (jogador).

2.2 ROTAÇÃO DA SETA (JOGADOR)

```
for j in range(pointer.length): #Rotation of the pointer

    ang+=1

    if (ang == 30):
        ang = 0

        vetorRotationPointer = -vetorRotationPointer

    pointer[j].rotate(axis=vetorRotationPointer, angle=omegaForArrow*dt)
```

Figura 4 - Rotação do Jogador

Aqui criou-se uma variável para controlar o ângulo da seta, assim, só precisamos de mudar o nosso vetor de rotação para obtermos o efeito pretendido.

2.3 COLISÕES

2.3.1 COLISÕES 2D

```
for k in range (balls.length): #Interseptions

        balls[k].pos = balls[k].pos + balls[k].velocity*dt

        for i in range(gears.length):

            #Collision with the Gears

            if ((balls[k].pos.x + balls[k].radius >= gears[i].pos.x - 1) and (bal

                if(score>=50):
                   score-=50

                balls[k].visible = False
                balls.splice(k,1)
                break

            if (balls[k].pos.y - balls[k].radius >= 20):
                balls[k].visible = False
                balls.splice(k,1)
                break

            if (balls[k].pos.y + balls[k].radius <= -15):
                balls[k].visible = False
                balls.splice(k,1)
                break

            if(balls[k].pos.x + balls[k].radius>=gears[i].pos.x + 2):
                score+=100
                balls[k].visible = False
                balls.splice(k,1)
                break
```

Figura 3 - COLISÕES 2D

Neste segmento, são verificadas as colisões entre as bolas (jogador) e as roldanas. Daí, só precisarmos de duas dimensões para verificarmos estes casos.

2.3.2 COLISÕES 3D

```
for l in range(enemies.length):

    #Collision with the enemies
    for k in range(balls.length):

        radSep = balls[k].pos - enemies[l].pos #Radius separation
        touchSep = (balls[k].radius + enemies[l].radius) * radSep.norm()

        if (radSep.mag < touchSep.mag): #Collision
            score-=500
            balls[k].visible = False
            balls.splice(k,1)
            break
```

Figura 4 - COLISÕES 3D

Nas colisões em três dimensões, aplicou-se uma forma que calcula o vetor de distância entre os dois centros (do inimigo e da bola do jogador). Depois calcula-se a separação, onde adicionando os dois raios e multiplicando pela normal do vetor calculado anteriormente, sabemos que existe interseção se o vetor de distância dos raios for menor que o cálculo anterior realizado (“touchSep”).

4 CONCLUSÃO

Este projeto foi uma proposta desafiante onde, sendo uma linguagem nova, propôs desafios para a concretização do mesmo.

Apesar de ser um jogo simples, é uma prova de conceito bem concedida.

Sendo assim, agradecemos ao professor José Campos Neves pela proposta e pelos conhecimentos transmitidos ao longo das aulas.

5 REFERÊNCIAS BIBLIOGRÁFICAS

http://www.glowscript.org/ -> Pequenas questões sobre código