Politécnico do Porto
Escola Superior de Media Artes e Design

Miguel Melo, 9150176
Rodrigo Queirós, 9170312

Totally Accurate Ball Game

Jogo 3d

Licenciatura em Tecnologias e Sistemas de Informação para a Web
Física Aplicada a Programação
Docente: Prof. Doutor José Neves

Vila do Conde, janeiro de 2019

Abstract

We were proposed to develop a three-dimensional game using Visual Python. Using a pre-existing example of billiards, we adapted and reformed it to meet our final game concept standards. The game concept was based on a popular childhood game called “Futebol sem bola” and on the popular market trend Battle Royal.

Keywords: Python, VPython, GlowScript, Physics, Game

Índice

Introdução

Este trabalho, realizado no âmbito da disciplina de Física Aplicada à Programação, tem como objetivo aplicar os princípios da física, tais como a colisão entre dois corpos, movimento linear e velocidade.

Recorrendo ao VPython foi possível programar um ambiente tridimensional para conceber o nosso "Totally Accurate Ball Game". Este jogo esta divido em dois modos completamente distintos. O primeiro, consiste em tentar chegar a uma área do campo adversário sem ser apanhado, visto que, é um jogo para duas pessoas o mesmo se dá para o adversário. O segundo modo, também para dois jogadores, consiste em empurrar o adversário para fora da área de jogo. Para vencer neste jogo, é necessário conseguir 5 pontos que podem ser adquiridos em qualquer um destes dois modos.

Foram várias as etapas que realizamos ao decorrer deste projeto, começamos por analisar os objetivos pretendidos com este trabalho e planear a sua conceção. Após termos uma estratégia definida começamos a juntar ideias para decidir o melhor caminho a seguir.

Este relatório está divido em três partes: Ideia, Desenvolvimento e Conclusão.

Ideia

Como o objetivo principal deste projeto é utilizar princípios da física aplicado a um ambiente tridimensional, era necessário arranjar uma ideia que nos permitisse desenvolver um jogo tendo em conta estes objetivos.

Para tal fizemos um brainstorming e este foi o resultado:

- Lançamento de um projétil para acertar em alvos em movimento, com um sistema de pontuação eficiente;
- Queda de uma bola que precisava de passar por buracos nas plataformas até chegar a área pretendida ou conseguir uma pontuação elevada;
- Queda aleatória de bolas onde precisávamos com o uso de um objeto dividir a bola para duas áreas distintas consoante a sua cor;
- Baseado em "Futebol sem bola", um jogo onde precisávamos de chegar ao limite do campo adversário sem sermos apanhados na sua área de jogo;
- Um jogo de sumo, onde o objetivo era empurrar o adversário para fora da arena, sendo que o adversário era controlado por inteligência artificial;
- Queda aleatória de bolas, onde o objetivo era acertar bolas de um certo formato e evitar bolas inimigas.

As limitações para o desenvolvimento deste jogo era muitas, mas a principal era a falta de tempo que nos impediu de seguir caminhos como o de programar um objeto a partir de inteligência artificial.

Tendo em conta que queríamos fazer o jogo o mais divertido possível e as limitações existentes as nossas escolhas ficaram divididas entre duas ideias.

- Baseado em "Futebol sem bola", um jogo onde precisávamos de chegar ao limite do campo adversário sem sermos apanhados na sua área de jogo;
- Um jogo de sumo, onde o objetivo era empurrar o adversário para fora da arena, sendo que o adversário era controlado por inteligência artificial;

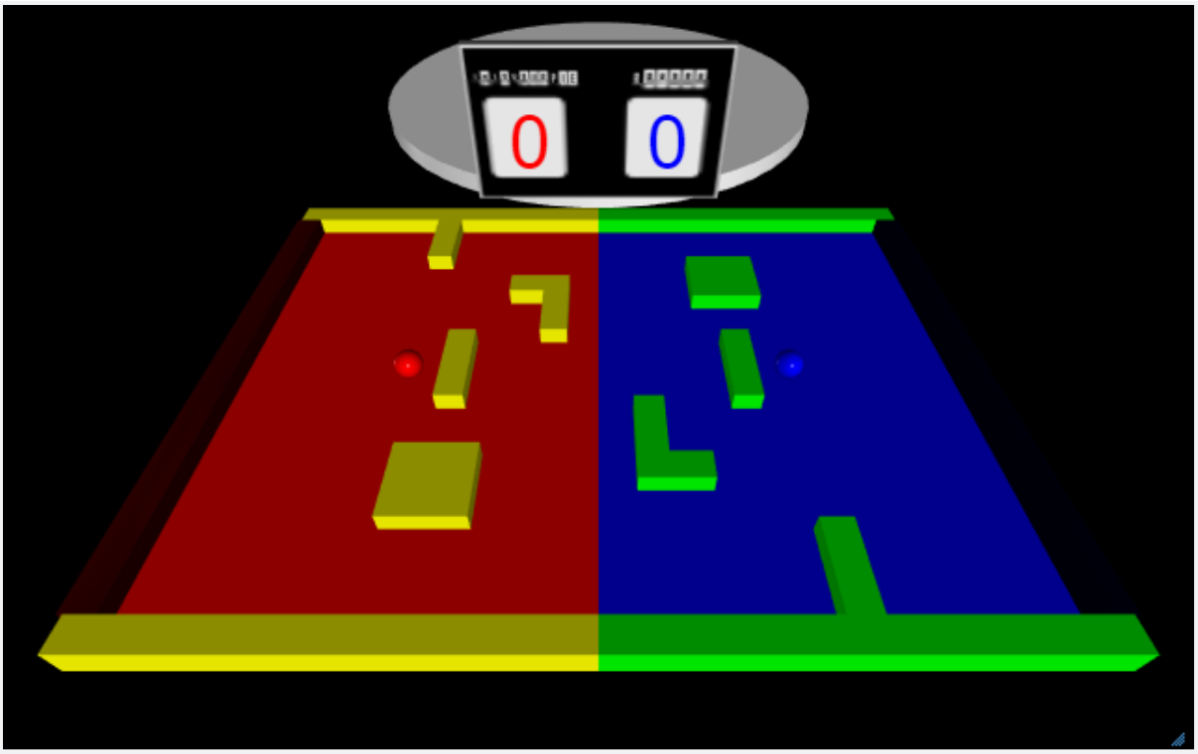
Como queríamos utilizar essas duas ideias e estas partilhavam elementos e mecânicas similares, acabamos por decidir escolher as duas, tendo sido necessário adaptar as ideias para obter um melhor resultado e coerência técnica.

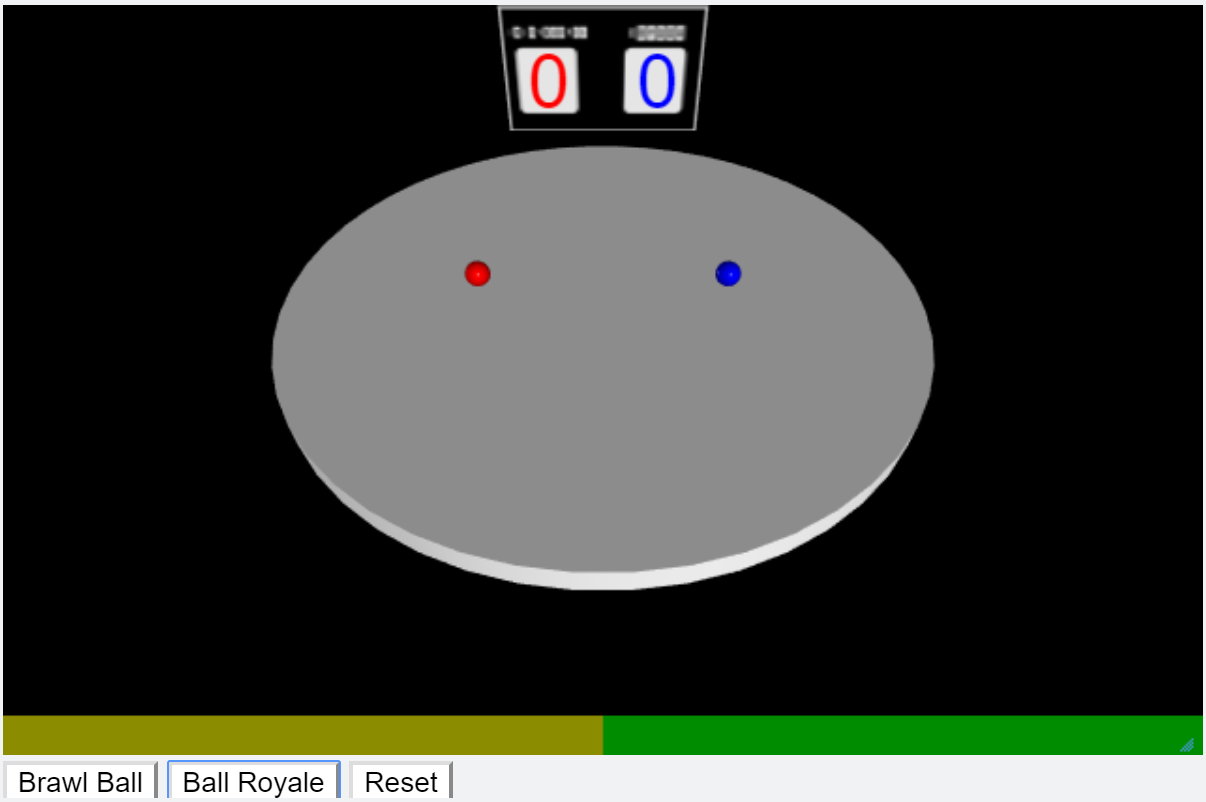
O resultado foi o seguinte:

Totally Acurrate Ball Game é um jogo de dois jogadores que permite a escolha entre dois modos diferentes, ganha o jogo o jogador que primeiro adquirir 5 pontos em qualquer um dos modos. O primeiro modo, Brawl Ball, é um jogo em que tens de ter atenção ao ataque e à defesa, visto que ganha ponto o jogador que chegar primeiro ao limite do campo adversário, mas também o jogador que capturar o adversário no seu campo. O segundo modo, Ball Royale, consiste em empurrar o adversário para fora da área de jogo, sendo essa área um círculo que vai diminuindo ao longo do tempo.

Desenvolvimento

Tendo a nossa ideia bem formulada e baseando-nos num programa já desenvolvido de bilhar o resultado foi o seguinte:





Este jogo engloba muitos conceitos físicos como colisões entre bolas e entre obstáculos, massa, velocidade, forças, entre outros que vão ser a seguir explicados.

```
scoreBoard = box(canvas = scene, size = 15*vec(2.4, 1.45, 0.1), texture = "https://i.imgur.com/3n1UAgM.png", color = color.white)
scoreBoard.pos = vec(0,10,-40)
bluePoints = 0
redPoints = 0
redScore = label(text = redPoints, pos = vec(-10, 6.5,-40), canvas = scene, opacity = 0, height = 40, color = color.red, box = False)
blueScore = label(text = bluePoints, pos = vec(10, 6.5, -40), canvas = scene, opacity = 0, height = 40, color = color.blue, box = False)
```

Com estas linhas de código conseguimos mostrar os pontos dos jogadores

```
#red side of the field
redField = box()
redField.size = vec(40,1,60)
redField.pos= vec(-20,0,0)
redField.color = color.red


circleField = cylinder()
circleField.radius = 40
circleField.axis = vec(0,3,0)
circleField.pos= vec(0,-3,-80)
circleField.color = color.white
#blue side of the field
blueField = box()
blueField.size = vec(40,1,60)
blueField.pos= vec(20,0,0)
blueField.color = color.blue
```

Os campos foram desenhados desta forma, sendo que o azul e o vermelho são pegados, fazendo parte do primeiro modo e o campo circular fazendo parte do segundo modo.

```
if(battleroyale == True):
  timecount +=1
  if (timecount > 100):
      circleField.radius -= 1
      timecount =0
```

Mais tarde foi adicionado ao campo circular um raio decrescente em função do tempo

```
blueBall = sphere()
blueBall.radius = 1.5
blueBall.pos = vec(InicialXBlue,2,InicialZBlue)
blueBall.color = color.blue
blueBall.force = arrow(axis = vec(0,0,0), shaftwidth = 0.2, color = color.blue)
blueBall.vel = vec(0,0,0) #Movimento incial X Z
blueBall.mass = 1.5 # 2.0

redBall = sphere()
redBall.radius = 1.5
redBall.pos = vec(InicialXRed,2,InicialZRed)
redBall.color = color.red
redBall.force = arrow(axis = vec(0,0,0), shaftwidth = 0.2, color = color.red)
redBall.vel = vec(0,0,0)
redBall.mass = 1.5

class pointerBluePos:
    x = blueBall.pos.x
    z = blueBall.pos.z

class pointerRedPos:
    x = redBall.pos.x
    z = redBall.pos.z
```

As bolas foram criadas com atributos como posição, cor, raio, força, velocidade e massa.

Também foi criado um pointer que mais tarde vai ser utilizado para mostrar a bola as posições que tem de seguir

```
#Parede vermelha
rail1 = box()
rail1.texture = {'file': "https://i.imgur.com/ss5OcnL.jpg"}
rail1.size = vec(3,3,55)
rail1.color = color.red
rail1.pos = vec(-39,1,0)

rail3 = box()
rail3.color = color.yellow
rail3.size = vec(40,3,3)
rail3.pos = vec(-20,1,29)

rail4 = box()
rail4.color = color.yellow
rail4.size = vec(40,3,3)
rail4.pos = vec(-20,1,-29)

rail8 = box()
rail8.color = color.yellow
rail8.size = vec(3,3,10)
rail8.pos = vec(-5,1,-10)
```

As paredes e obstáculos foram todas criadas a mão sendo mais tarde adicionadas a um array.

```
rail = []
rail.append(rail1, rail2, rail3, rail4,rail5,rail6,rail7,

ball = []
ball.append(blueBall, redBall)
```

```
game1  = button(bind = click, text = 'Brawl Ball')
def click():
    nonlocal battleroyale
    blueBall.pos.x = InicialXBlue
    blueBall.pos.z = InicialZBlue
    redBall.pos.x = InicialXRed
    redBall.pos.z = InicialZRed
    pointerBluePos.x = InicialXBlue
    pointerBluePos.z = InicialZBlue
    pointerRedPos.x = InicialXRed
    pointerRedPos.z = InicialZRed
    blueBall.vel.x = 0
    blueBall.vel.z = 0
    redBall.vel.x = 0
    redBall.vel.z = 0
    scene.range = 35
```

Foram criados 3 botões dos quais dois para mudar de modo e uma para dar reset ao jogo. O código dos botões é semelhante, modificar posições da bola e da câmara.

```
def playerMovement(event):
    ## FUCK NON LOCAL SHIT MOFO PLS
    nonlocal playerOneMoveRight, playerOne
    if event.which == 83:
        playerTwoMoveRight = True
    if event.which == 87:
        playerTwoMoveLeft = True
    if event.which == 68:
        playerTwoMoveForward = True
    if event.which == 65:
        playerTwoMoveBack = True

    if event.which == 37:
        playerOneMoveForward = True
    if event.which == 39:
        playerOneMoveBack = True
    if event.which == 40:
```

A leitura de eventos do teclado foi feita desta forma

```
#Collision with walls
  for i in range(len(ball)):
     for j in range(len(rail)):
        #Collision redB with bluewall
        if(redBall.pos.x + redBall.radius >= 37 and redBall.pos.z - redBall.radius > -40):
            blueBall.pos.x = InicialXBlue
            blueBall.pos.z = InicialZBlue
            redBall.pos.x = InicialXRed
            redBall.pos.z = InicialZRed
            pointerBluePos.x = InicialXBlue
            pointerBluePos.z = InicialZBlue
```

Depois está tudo dentro de um ciclo while cheio de colisões, incrementos e cálculos de diferenças e velocidades.

```
            pointerRedPos.z -= 1
        if playerTwoMoveRight:
            pointerRedPos.z += 1
        if playerTwoMoveBack:
            pointerRedPos.x -= 1

        blueBall.force.axis.x = pointerBluePos.x - blueBall.pos.x
        blueBall.force.axis.z = pointerBluePos.z - blueBall.pos.z
        blueBall.force.pos = blueBall.pos
        blueBall.force.shaftwidth = 0.5
        blueBall.force.axis.y = 0
```

Conclusão

Após o trabalho desenvolvido foram tiradas algumas conclusões em relação ao estado final do projeto. Uma destas seria do aperfeiçoamento das colisões, relativamente as colisões das bolas com os cantos dos obstáculos e do campo de jogo, a utilização de texturas de maior qualidade e o desenvolvimento de uma interface gráfica mais apelativa, e por último a implementação de um menu principal.

Com o passar do tempo foi possível aprimorar as capacidades relacionadas não só com VPython e Python, mas também sobre física e programação no seu todo. O conhecimento sobre variáveis não globais em Python e os problemas que estas criaram durante o desenvolvimento do jogo foram algo que surgiu várias vezes, mas que ultimamente, acabaram suprimidas.

Futuramente é esperado mais eficiência com as ferramentas usadas e menos tempo perdido na resolução dos problemas que se revelam, e quem sabe, o melhoramento do jogo com todas as sugestões antes destacadas e outras que possam surgir.

Referencias bibliográficas

"geometry - Circle-Rectangle collision detection (intersection)," *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/401847/circle-rectangle-collision-detection-intersection. [Accessed: 26-Jan-2019].

"overview." [Online]. Available: http://www.glowscript.org/docs/GlowScriptDocs/index.html. [Accessed: 26-Jan-2019].

"VPython Help." [Online]. Available: https://vpython.org/contents/docs/. [Accessed: 26-Jan-2019].

"Physics through GlowScript - an Introductory Course," *trinket.io*. [Online]. Available: https://bphilhour.trinket.io/physics-through-glowscript-an-introductory-course. [Accessed: 26-Jan-2019].