Maze Runner

Jogo de Labirinto Implementado com Deteção de Cor e Posicionamento Através de um DualShock

Rui Ramos up201705782
Rúben Lôpo up201709326

Visão Computacional

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Porto, Portugal
Janeiro 2020

Índice

1 Apresentação da Ideia **2**

2 Tecnologias Estudadas **3**

3 Estratégias de Implementação **3**

4 Conclusão **6**

1 Apresentação da Ideia

No âmbito da Unidade Curricular Visão Computacional, foi-nos proposto desenvolver um projeto, de tema livre, que abrangesse um dos campos de estudo da área referente.

Depois de alguma pesquisa e *brainstorming*, surgiu a ideia de desenvolver um jogo em que cada ação consistisse numa **deteção de cor e posicionamento**, a partir de uma câmara com um ***DualShock*** desenhado. Levantaram-se então várias possibilidades de jogo a ser implementado, tendo sido escolhido o **Jogo do Labirinto**, por uma questão de afinidade relativamente à infância dos elementos do grupo.

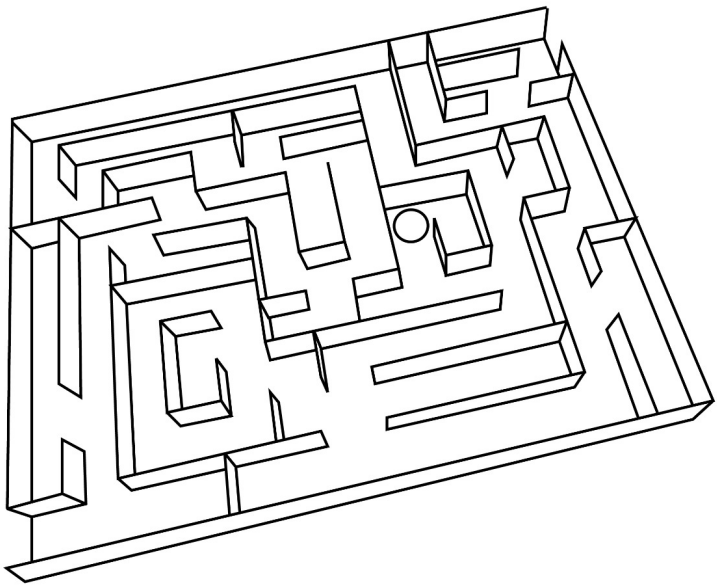

Figura 1: Jogo do Labirinto

2 Tecnologias Estudadas

Para a implementação deste projeto, inicialmente foi pensada uma implementação em *Java*, porém as bibliotecas de Visão Computacional desta tecnologia envolviam a instalação e configuração de outros *plugins* adicionais e, por uma questão prática, foi decidido utilizar a linguagem de programação ***Python***, na versão 3.9.0.

Para implementação de várias funcionalidades, foram utilizados os seguintes pacotes:

- ***OpenCV***: Biblioteca concebida para resolver problemas de visão computacional. Esta ferramenta faz uso da Numpy, que é uma biblioteca altamente otimizada para operações numéricas, com uma sintaxe ao estilo MATLAB. Através desta, captamos imagens, de forma a ser possível analisar os *frames* continuamente.
- ***Random***: Biblioteca utilizada para gerar números aleatórios numa fase de testagem.
- ***VLC***: Biblioteca utilizada para reprodução de música.
- ***Time***: Biblioteca utilizada para criar tempos de espera na execução do jogo.
- ***Colored***: Biblioteca utilizada para modificar a cor de texto.

3 Estratégias de Implementação

Para o bom funcionamento do projeto, decidimos planear a implementação de forma antecipada e faseada.

Numa primeira fase, foi implementado o labirinto que consiste numa estrutura de dados multidimensional de tamanho fixo, com métodos auxiliares para construção, possibilidade de jogada, jogada aleatória e impressão.

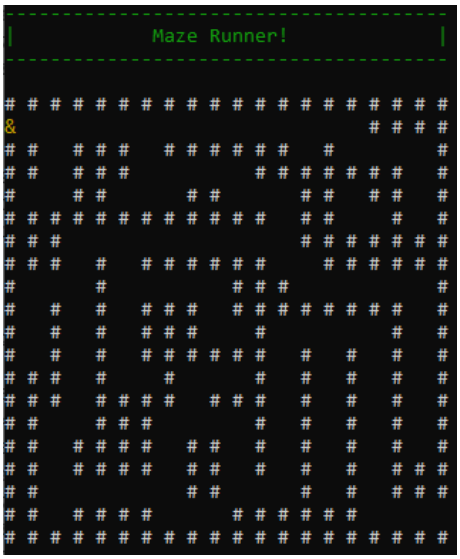

Figura 2: Tabuleiro de Jogo

Como o principal objetivo do projeto está relacionado com a captação e análise de imagem, decidimos formatar os labirintos como estáticos e previamente construídos.

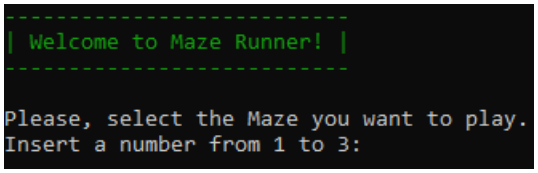

Figura 3: Menu de Seleção de Tabuleiro

Para a captura e análise de imagem, foi desenhado um ***DualShock*** na imagem apresentada. O utilizador, de forma a mover-se no tabuleiro, deve utilizar um objeto de cor definida e colocá-lo numa das figuras desenhadas, de forma à cor e presença do mesmo serem analisadas, e efetuada a jogada.

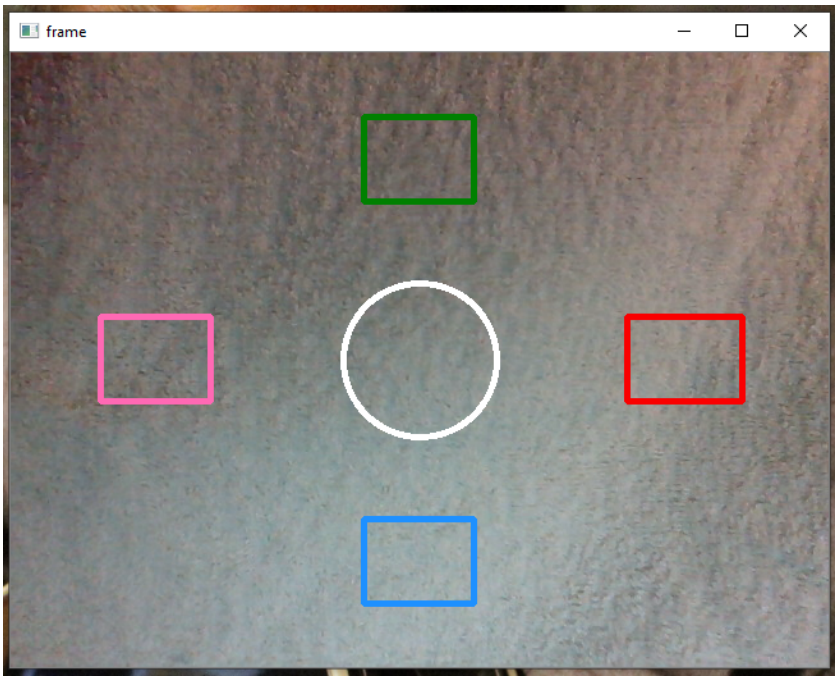

Figura 4: Imagem captada com DualShock desenhado

De forma a proceder à verificação da posição selecionada pelo jogador, é efetuada uma contagem de *pixeis*, nas regiões correspondentes, cuja cor se encaixe num intervalo previamente definido. A região com mais *pixeis* associados ao intervalo *RGB* em questão é a então selecionada para a jogada. Foi definido um número mínimo de *pixeis* aceitável, de forma a evitar ruído dos dados captados.

```
def checkPlay():

    # rec2_pt1 = (482, 206)
    # rec2_pt2 = (572, 272)
    correctPixelCountRight = 0;
    for i in range(482, 572):
        for j in range(206, 272):
            (b, g, r) = frame[j, i]
            if b<=50 and g<=50 and r>=110:
                correctPixelCountRight = correctPixelCountRight + 1;

    # rec1_pt1 = (70, 206)
    # rec1_pt2 = (156, 272)
    correctPixelCountLeft = 0;
    for i in range(70, 156):
        for j in range(206, 272):
            (b, g, r) = frame[j, i]
            if b<=50 and g<=50 and r>=110:
                correctPixelCountLeft = correctPixelCountLeft + 1;

    # rec4_pt1 = (276, 364)
    # rec4_pt2 = (362, 430)
    correctPixelCountDown = 0;
    for i in range(276, 362):
        for j in range(364, 430):
            (b, g, r) = frame[j, i]
            if b<=50 and g<=50 and r>=110:
                correctPixelCountDown = correctPixelCountDown + 1;

    # rec3_pt1 = (276, 50)
    # rec3_pt2 = (362, 116)
    correctPixelCountUp = 0;
    for i in range(276, 362):
        for j in range(50, 116):
            (b, g, r) = frame[j, i]
            if b<=50 and g<=50 and r>=110:
                correctPixelCountUp = correctPixelCountUp + 1;

    maxPixels = max(correctPixelCountRight, correctPixelCountLeft, correctPixelCountDown, correctPixelCountUp);
    if(maxPixels==correctPixelCountRight and maxPixels>800):
        return "D"
    if(maxPixels==correctPixelCountLeft and maxPixels>800):
        return "A"
    if(maxPixels==correctPixelCountDown and maxPixels>800):
        return "S"
    if(maxPixels==correctPixelCountUp and maxPixels>800):
        return "W"
```

Figura 5: Função analisadora de Imagem

Quando o jogador alcança a posição de chegada, é reproduzida uma música de vitória.

4 Conclusão

Concluído o projeto, afirmamos que o objetivo foi cumprido. A ideia inicial foi implementada tal como planeado previamente e o resultado final é do agrado de todo o grupo de trabalho.