Instituto Politécnico de Viana do Castelo

Escola Superior
de Tecnologia
e Gestão

PROCESSING-RETRO GAME

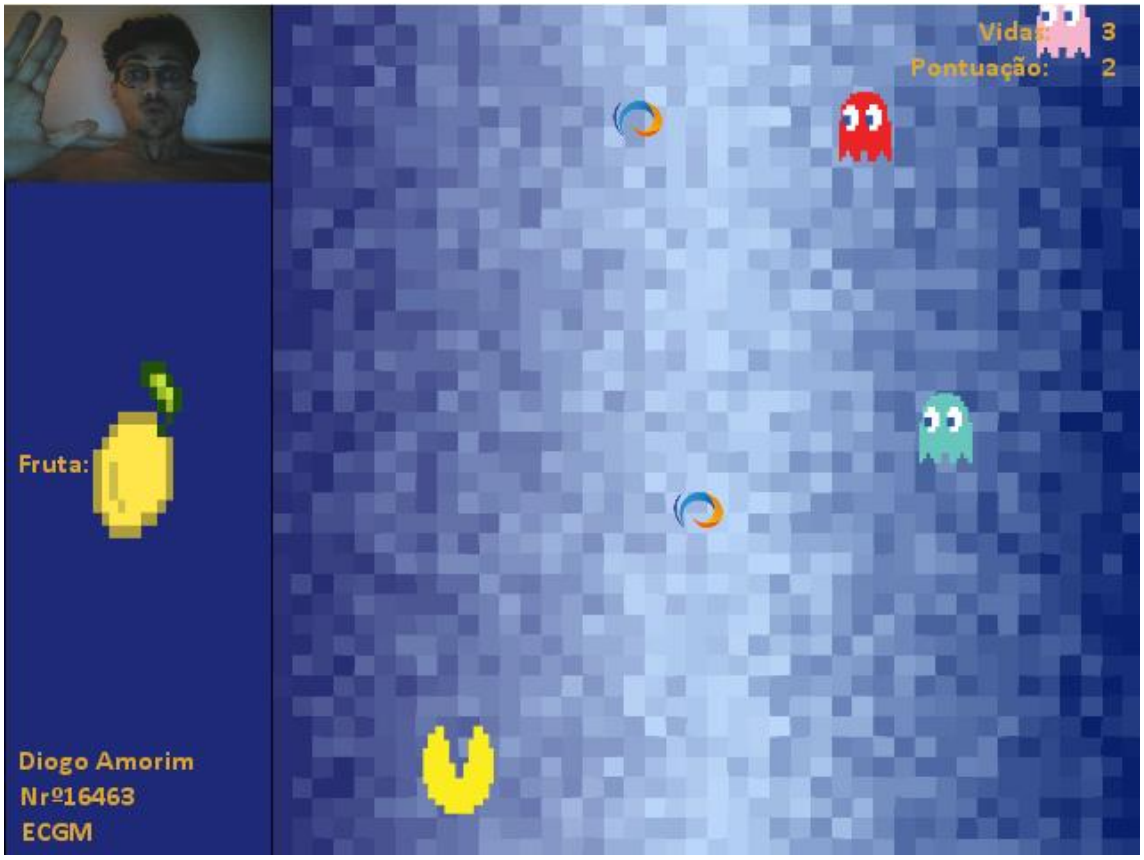
DIOGO MIGUEL ALMEIDA AMORIM Nº:18463

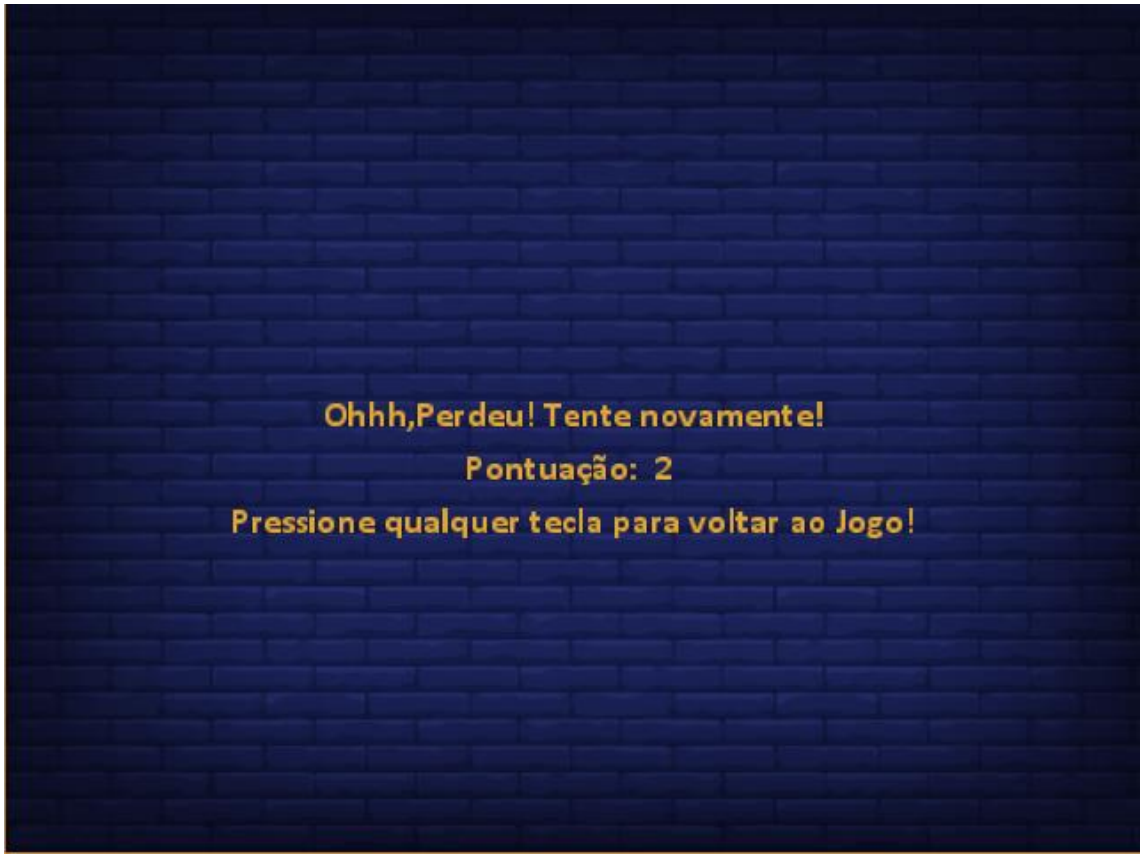
Objetivo

Este jogo foi uma modificação de um jogo mundialmente famoso chamado “PAC MAN”. Nesta minha modernização do mesmo o *player* têm de eliminar os seus inimigos com frutas que são uma espécie de “balas” e colecionar o máximo número de logotipos de ECGM.

Cada inimigo que o jogador deixa passar, o jogador perde uma vida. Por cada logotipo que consegue colecionar, ganha um ponto.

O jogador perde o jogo quando não tiver nenhuma vida.


Vida: 3
Pontuação: 2
Fruta:
Diogo Amorim
Nrº16463
ECGM


Ohhh,Perdeu! Tente novamente!
Pontuação: 2
Pressione qualquer tecla para voltar ao Jogo!

Funcionalidades

Movimento

A forma como se deteta o movimento é através da comparação de duas imagens (*frames*) da captura de vídeo. Grava-se sempre a imagem anterior à imagem atual da captura do vídeo. Com estas duas imagens pode-se fazer a comparação dos pixéis, através da cor deles, de cada uma delas. Caso a diferença de cor seja superior a um dado limite, grava-se a posição desse pixel.

No final, calcula-se a média das posições dos pixéis que tinham diferenças de cor superiores a esse limite. Essa posição média é depois passada para a posição do retângulo que o jogador controla. Isto tudo é conseguido através de uma biblioteca, disponível no *Processing*, que acede à webcam do computador de modo a que capturar vídeo.

```
for(int x = 0; x < video.width; x++)  {
  for(int y = 0; y < video.height; y++)    {
    int loc = x + y * video.width; //indice de cada pixel
    //--inverter a imagem na horizontal
    int loc2 = (video.width - x - 1) + y * video.width;
    pixels[loc2] = video.pixels[loc];
    color current = video.pixels[loc];//vai buscar o pixel da captura de video que está na posição loc
    color previous = prevFrame.pixels[loc];//vai buscar o pixel da imagem armazenada que está na posição loc
    //armazena a cor de cada pixel
    float r1 = red(current);
    float g1 = green(current);
    float b1 = blue(current);
    float r2 = red(previous);
    float g2 = green(previous);
    float b2 = blue(previous);
    float diff = dist(r1, g1, b1, r2, g2, b2);//faz a diferença de cor entre um pixel e outro

    if(diff > threshold)   {
      moveX += x;
      mediaMove++;  } } }
updatePixels();//recarrega todos os pixeis da imagem
if(mediaMove != 0)//faz a posição média dos pixeis guardados
{
  moveX = moveX / mediaMove;
}

if(moveX > posX + mexePixels / 2)
{
  posX += mexePixels;
}
else if(moveX < posX - mexePixels / 2)
{
  posX -= mexePixels;
}
```

Som

A captura do som é feita através de uma biblioteca que recorre ao microfone do computador e regista o áudio. A interação foi feita de modo a que fosse armazenada a amplitude do som capturado, servindo depois para comparar essa amplitude com um limite estabelecido. Caso superasse esse limite, dispara uma fruta.

```
inputAudio = new AudioIn(this, 0);

inputAudio.start();

aAmplitude = new Amplitude(this);

aAmplitude.input(inputAudio);
```

```
float volume = aAmplitude.analyze();//regista a amplitude do som captado
float limite = 0.1;

// amplitude maior limite cria uma fruta
if(volume > limite && millis() > time + 500)
{
  Frutas bal = new Frutas(int(posPlayerX), int(posPlayerY), 8, tiposfrut);
  frut.add(bal);
  tiposfrut = criaTipo();
  somMissil.play(0);
  time = millis();
}
```

Colisões

As colisões são feitas através de uma função que calcula a distância do centro do até à fronteira para cada um dos dois retângulos. Caso os valores sejam iguais, a função retorna verdadeiro, caso não sejam, a função retorna falso. Esta verificação é feita para os quatro lados de cada retângulo.

```
//retorna verdadeiro se ouver colisão entre um fruta e um inimigo

boolean rectRect(int x1, int y1, int w1, int h1, int x2, int y2, int w2, int h2) {


  if (x1+w1/2 >= x2-w2/2 && x1-w1/2 <= x2+w2/2 && y1+h1/2 >= y2-h2/2 && y1-h1/2 <= y2+h2/2) {
    return true;
  }
  else {
    return false;
  }
}

 //colisão fruta inimigo, elimina
 for(int k = 0; k < frut.size(); k++)
 {
   Frutas b = frut.get(k);
   if(rectRect(b.posfrutaX, b.posfrutaY, b.rectW, b.rectH, i.x, i.y, int(i.w), int(i.h)) )
   {
     inimigo.remove(j);
     frut.remove(k);

     break;
   }
```

Classes

As classes usadas para este jogo, são as seguintes:

- RETROGAME – construção do jogo
- Frutas – tiros de frutas;
- Inimigos – alocado todo o tipo de inimigos;
- ECGM - logotipos ECGM que dão pontos;
- Jogador – alocado o jogador PAC-MAN;

Regras Jogo

O jogador começa cada jogo, com sete vidas e pontuação zero. Para poder pontuar, o jogador tem que colecionar os logotipos da ECGM. Por cada logotipo colecionado, o jogador ganha um ponto. Ao mesmo tempo tem que eliminar os inimigos com as frutas. Caso os inimigos escampem, perde uma vida. Quando a vida chegar a zero, perde o jogo.

Conclusão

Poderia ter feito melhorias, mas o jogo reúne as características necessárias para criar um bom momento de entretenimento.

Podiam ser feitas melhorias na deteção de movimento e no próprio visual para que o jogo fosse mais eficaz.

Referências

freepik. (s.d.). *https://www.freepik.com/index.php?goto=2&searchform=1&k=PIXEL+ART.*

http://www.classicgaming.cc/classics/pac-man/sounds. (s.d.). Obtido de classicgaming.

http://www.classicgaming.cc/classics/pac-man/vector-art. (s.d.). Obtido de classicgaming.

processing. (s.d.). *https://processing.org/reference/libraries/*.