

O Duelo

Mario de Castro - Humberto Vieira - William de Alvelos
São Paulo - SP, Brasil

Abstract

Para a disciplina de Projeto Integrador III foi proposto o desenvolvimento de um jogo que utilize visão computacional como principal meio de interação do usuário. O jogo proposto, chamado "O Duelo", consiste em um jogo de ação para dois jogadores que, utiliza duas câmeras para captar o movimento dos mesmos separadamente, buscando simular a tensão de um duelo ao estilo velho oeste. O projeto foi escrito na linguagem C e utiliza a biblioteca gráfica Allegro5 assim como a biblioteca de visão computacional OpenCV. O seguinte artigo busca detalhar o processo de desenvolvimento do projeto com ênfase nos algoritmos de visão computacional utilizados, assim como ressaltar dificuldades e desafios encontrados durante todas as etapas.

Palavras Chave: Visão Computacional. OpenCV. Velho Oeste. Duelo.

1 Introdução

O jogo aqui descrito foi elaborado para disciplina de Projeto Integrador III, ministrada pelo Prof. Marcelo Hashimoto no terceiro semestre do curso de Bacharelado em Ciência da Computação do Centro Universitário Senac.

Foi requerido aos alunos da disciplina o desenvolvimento de um jogo na linguagem de programação C, em conjunto da biblioteca gráfica Allegro5, que utilizasse visão computacional, através da biblioteca OpenCV, como principal meio de interação do jogador.

Para o trabalho descrito neste artigo o grupo tinha como objetivo criar um jogo para dois jogadores na qual a visão computacional fosse utilizada para captar o movimento de um jogador, utilizando este movimento de forma passiva, com o objetivo de simular a tensão de um duelo ao estilo velho oeste.

O Duelo é composto por 6 telas, sendo elas, a tela inicial, que mostrará o menu principal, a partir dela poderão ser acessadas a tela de créditos, que mostrará os responsáveis pelo desenvolvimento do jogo e uma tela de tutorial que dará instruções de como jogar. Ao clicar em iniciar jogo, o jogador

entrará em um menu no qual poderá escolher entre 1, 3 ou 5 rodadas que serão jogadas no máximo para que apenas um jogador saia vencedor do duelo. Este menu por sua vez, será feito através de visão computacional, uma vez que o jogador 1, ao mexer sua arma, poderá escolher entre as opções e iniciar o jogo. Além destas, há a tela de resultado do duelo, na qual ela também pode decidir se jogará novamente ou não.



Figure 1: Tela de inicial



Figure 2: Tela de rodadas

Ao escolher a quantidade de rodadas, antes de iniciar o duelo, o jogador deve permanecer imóvel em frente ao adversário até receber um aviso do jogo que ele pode se mover. Para verificar o movimento o jogo utiliza duas câmeras para captar as interações dos dois jogadores separadamente.

Ao iniciar a partida, o jogador deve permanecer imóvel de frente para o adversário até receber um aviso do jogo que ele pode se mover. As duas câmeras devem estar entre os jogadores, cada uma apontada para um jogador.

Ao escutar o aviso os jogadores devem reagir o mais rápido possível para

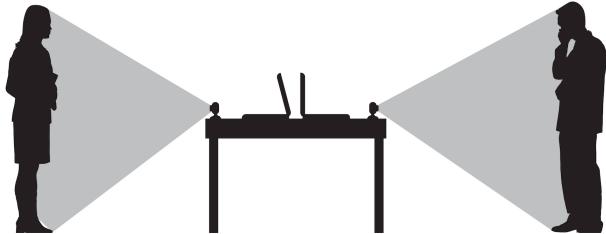


Figure 3: Posição dos jogadores

apontar uma réplica de arma segurada por eles para o adversário e atirar. As réplicas das armas são feitas de papel machê, que tem uma ponta pintada de amarelo que é usada para detecção do tiro.

Assim, permite que o algoritmo do jogo detecte o disparo da arma, através da detecção da cor. O jogador que atirar primeiro após o aviso ganhará a rodada. Além disso, quem se mover antes do aviso perderá a rodada.

Durante a partida, os jogadores poderão receber *feedbacks* visuais de dados do duelo como:

- Qual round está;
- O placar da partida;
- A hora de atirar;
- Vencedor do round;
- Vencedor do duelo;
- Quando é possível se mover;

No jogo são utilizados dois algoritmos de visão computacional principais: um para detecção do movimento do jogador e outro para reconhecimento da cor que estão na ponta das réplicas das armas. Os dois algoritmos serão descritos em maiores detalhes na próxima seção.

2 Desenvolvimento

2.1 Detecção de Movimento

Para realizar a detecção de movimento dos jogadores utilizou-se uma versão de um algoritmo que compara o frame atual da câmera com o frame imediatamente anterior em busca de diferenças entre seus pixels. Caso a diferença seja grande o suficiente o movimento é registrado.

Ao iniciar o jogo o primeiro frame da câmera é salvo em uma matriz de vetores *matriz1*, onde cada vetor representa os valores RGB de cada pixel da imagem. No decorrer do jogo cada novo frame da câmera é então salvo em uma segunda matriz de vetores RGB *matriz2*.

Em seguida o algoritmo percorre todos os campos de cada matriz e compara os valores R (vermelho), G (verde) e B (azul) de cada vetor da *matriz1* com seu vetor correspondente da *matriz2*. Caso o valor de algum dos campos R, G ou B do vetor da primeira matriz difira do valor RGB da segunda matriz para mais ou para menos de que um certo *RANGE* previamente definido considera-se que houve mudança naquele pixel. Por exemplo, caso seja definido que o *RANGE* é igual a 20 e o valor R de um pixel da *matriz1* é igual a 100, só será identificado que houve mudança no pixel se o valor do pixel correspondente na *matriz2* for maior que 120 ou menor que 80.



Figure 4: Visualização dos pixels diferentes

Para cada pixel definido como diferente é somado 1 a um contador de pixels diferentes q que é iniciado com valor 0, e ao final do algoritmo ter percorrido toda a matriz terá a quantidade de pixels diferentes entre as duas matrizes. Caso essa quantidade q seja maior que uma valor *sensibilidade* a ser definido, é considerado que houve movimento entre os frames. O valor da *sensibilidade* é determinado pela divisão entre quantidade de pixels $qpixels$

da matriz por um valor *razão* pré definido. de forma que *sensibilidade* = *qpixels* / *razão*. quanto maior o valor de *razão*, menor será a quantidade de pixels diferentes necessários para identificação de movimento. Então identificaremos o movimento se:

$$q > \frac{qpixels}{razao}$$

A precisão do algoritmo pode ser facilmente controlada alterando-se o valor do *RANGE* que determina se houve mudança em um pixel ou o valor de *razão*, que determina a quantidade de pixels que precisam ser diferentes para que o movimento seja identificado.

2.2 Detecção de Cor

Para a detecção do tiro do jogador foi criado um algoritmo para detecção de cor por meio do sistema de cores hsv. Esse sistema foi escolhido por simplificar identificação da matriz de uma cor de três variáveis no sistema RGB para somente uma no sistema hsv.

Como a matriz utilizada pelo algoritmo fornece os valores no sistema RGB foi necessário fazer a conversão, que ocorre da seguinte maneira. Primeiro dividesse os valores de R, G e B por 255:

$$\begin{aligned} R' &= \frac{R}{255} \\ G' &= \frac{G}{255} \\ B' &= \frac{B}{255} \end{aligned}$$

Depois calcula-se os valores máximos (Cmax) e mínimos (Cmin) entre R', G' e B', assim como a diferença Δ entre esses valores:

$$\Delta = Cmax - Cmin$$

Feito isso determinasse o valor da matriz (H) de acordo com o valor de Cmax.

Se $Cmax = R'$:

$$H = 60 * \left(\frac{G' - B'}{\Delta} \right) mod 6$$

Se $Cmax = G'$:

$$H = 60 * \left(\frac{B' - R'}{\Delta} + 2 \right)$$

Se $Cmax = B'$:

$$H = 60 * \left(\frac{R' - G'}{\Delta} + 4 \right)$$

O valor do iluminação (V) é igual ao valor de $Cmax$ ($V = Cmax$). O valor da saturação não foi calculado pelo algoritmo por não ser necessário para a detecção de uma matiz específica.

Tendo o valor da matiz (H) podesse identificar quantos pixels da tela possuem uma matiz específica de acordo com quantos pixels tem o valor de H perto de um certo *range*. De forma análoga à detecção de movimento, determina-se se há a matiz procurada na tela de a quantidade de pixels com essa matiz (q) for maior que a quantidade de pixels da tela ($qpixels$) dividido por uma valor (sensibilidade) pré definido.

$$q = \frac{qpixels}{sensibilidade}$$

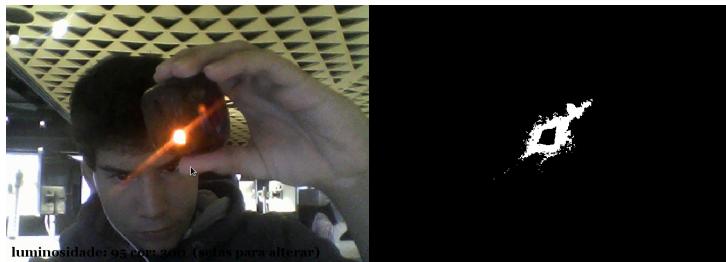


Figure 5: Visualização dos pixels com cor vermelha

3 Considerações Finais

A partir deste projeto conclui-se que OpenCV é uma ferramenta bastante poderosa para desenvolvimento de aplicações relacionadas a visão computacional. Entretanto, ainda é mais importante saber como esses algoritmos funcionam, como na detecção de objetos e formas, na detecção de movimento e também na detecção de cores em diferentes ambientes. A quantidade de fatores que deve ser levado em conta para que o algoritmo seja cada vez mais preciso é outro dos principais aprendizados, uma vez que ao tratá-los o jogo fica mais adaptável possível, sem precisar alterá-lo no código-fonte. Portanto, visão computacional é um assunto introduzido através deste projeto que abrirá portas para estudos mais profundos de algoritmos e aplicações dos mesmos em soluções de problemas na realidade.