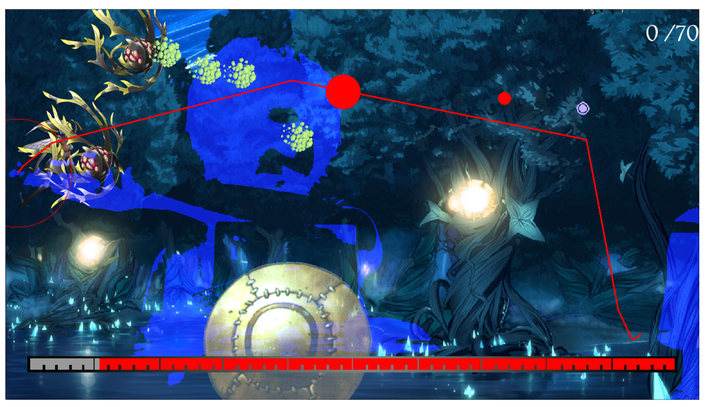
Dragon Souls of Senac: Touhou Edition

Gabriel Nopper Silva, Danilo Makoto Ikuta

Bacharelado em Ciência da Computação
Centro Universitário SENAC - Campus Santo Amaro (SENAC-SP)
Av. Engenheiro Eusébio Stevaux, 823 – Santo Amaro, São Paulo – CEP 04696-000 – SP – Brasil
(ganopper@gmail.com, danilo0195@hotmail.com)

1. Introducão

ESTE terceiro Projeto Interativo do Bacharelado de Ciência da Computação traz a ideia da visão computacional. Neste relatório explica-se como com apenas linguagem C, auxiliada das bibliotecas OpenCV e Allegro5, foi possivel criar um jogo de luta com escudo e espada, utilizando como unico controle usuário-maquina uma camera, e dois emissores de luz, uma vermelha e outra verde.



2. Objetivos

O objetivo do projeto é desenvolver um jogo em Linguagem C, utilizando apenas duas bibliotecas externas - OpenCV, referente apenas à captura de imagem (a segmentação das imagens da câmera ficou à cargo dos alunos), e Allegro5, referente à imagem. O jogo aqui descrito é um jogo de ação, cujo jogador luta contra vários monstros, atacando suas imagens com uma espada(Que trata-se de uma projeção de luz vermelha), por diversas ”Fases”, cada uma com monstros diferentes que lançam ataques na tela que podem ser bloqueados por um escudo (Que trata-se da projeção de luz verde). Tem-se como principais objetivos, criar uma detecção rapida e fiel ao movimento do corte com luz vermelha, e da defesa com luz verde, e como objetivos secundários, estruturar um jogo estável, divertido, e com um fluxo rápido. Para poder identificar essas luzes, o primeiro passo foi converter o espaço de cor de RGB(vermelho, verde, azul) padrão para HSV(matiz, saturação, valor), visto que os testes para identificar apenas a cor verde ou vermelha em RGB apresentavam resultados consideravelmente diferentes quando se trata de pequenas variações na iluminação, que pode ser mais facilmente ajustado no espaço HSV. Durante o processamento de câmera, este se tornou o espaço padrão do projeto. Outro ponto foi utilizar fontes de luz das cores vermelha e verde ao invés de simplesmente um objeto da cor para evitar que a camiseta de um jogador, por exemplo, seja acidentalmente detectada pelo programa. Além disso, o jogo exibe a silhueta do jogador(em um tom azulado), fazendo uso do método de subração de fundo. No momento em que o programa é inicializado, ele passa um certo período (sem que o jogador apareca na câmera) detectando diversas vezes o fundo(background) e a cada captura é feita uma média dos valores para cada pixel, afim de produzir uma imagem referencial mais estável, que durante o ciclo de câmera, cada pixel é comparado com seu correspondente no quadro(frame) mais atual da câmera. Dependendo da variação nos valores em HSV, um certo pixel pode ser considerado como de fundo(background) ou pertencente ao jogador.

O algoritmo de obtenção de fundo usado foi:

```
Obtenção de fundo{
    //Obtenção do HSV
    Dado certo número de vezes{
        Para cada pixel{
            Converte os valores RGB para HSV

            Soma os valores à imagem de saida
        }
    }

    Para cada pixel{
        Divide o valor pela quantidade de medições
    }
}

Ciclo de Câmera{
    Para cada pixel da imagem de entrada{
        Converte para HSV o pixel

        Compara os valores HSV com os parâmetros para a espada

        Se(For um pixel da espada){
            Soma as coordenadas (x, y) à respectiva variável
            Incrementa o contador de pontos da espada
        }

        Compara os valores HSV com os parâmetros para
          personagem

        Se(For um pixel de personagem){
            Marca como ponto de personagem em uma matriz
              auxiliar
        }

        Senão{
            Marca como ponto de fundo na matriz auxiliar
        }

        Identificação de escudo
        //Realiza as mesmas operações da identificação da
          espada, mas com parâmetros e variáveis diferentes
    }

    Se(Número de pontos de escudo for maior que um certo nú
      mero){
        Calcula as coordenadas(x, y) do centro de massa dos
          pontos do escudo
    }

    Senão{
        Atribui valores inválidos às coordenadas do centro de
          massa, indicando que não há escudo
    }

    Se(Número de pontos de espada for maior que dado número){
        Calcula coordenadas do centro de massa dos pontos da
          espada
        Insere essas coordenadas numa fila
    }

    Senão{
        Insere as coordenadas do ponto mais recente da fila,
          se houver.
    }

    Se(Número de pontos na fila for maior que um certo valor){
        Retira um ponto da fila.
    }

    Percorre a fila, desenhando linhas entre os pontos
      consecutivos, representando o ”rastro da espada”.
}
```

3. Metodologia

O jogo propriamente dito roda através de um ciclo principal que se encerra apenas no caso do jogador perder ou ganhar o jogo. Dentro deste ciclo estão diversos outros pequenos ciclos, que fazem o programa executar. Em forma de pseudocódigo, a estrutura geral do ciclo principal ficaria da seguinte forma:

```
Enquanto(Ultimo chefe não derrotado){

    // Ciclo dos monstros
    Para (cada monstro na lista){
        Realiza Movimento;
        Avança contador de timer p/ ataque;
        Se (contador de ataque pronto){
            Gera ataque;
            Reseta contador de ataque;
        }
        Desenha bitmap do monstro;
    }

    Verifica e limpa monstros mortos;

    // Ciclo do chefe
    Se (Há um chefe na lista){
        Se (Vida abaixo de 0){
            Verifica e limpa monstros;
        }
        Se não{
            Executa movimento;
            Avança cont. de timer p/ ataque;
            Se (Contador de ataque pronto){
                Gera ataque;
                Reseta contador de ataque;
            }
        Desenha bitmap do chefe;
        }
    }

    //Ciclo de câmera
    Chama captura da câmera;

    //Ciclo de ataques
    Para(Cada ataque na lista){
        Move ataque;
        Se(Pos. do Ataque for a Pos. alvo){
            Se(acertou escudo){
                Executa audio;
                Deleta ataque;
            }
            Se não{
                Subtrai dano da defesa;
                Causa Dano;
            }
        }
    }

    //Ciclo de interface

    Se(Há animações para fazer)
        Executa animações;

    Desenha escudo;
    Desenha Elementos de interface;

    Joga elementos desenhados na tela;

    //Ciclo do jogador
    Para (Cada monstro na lista){
        Verifica se foi acertado;
        //Dentro desta verificação ocorre dano
    }

    //ciclo de fim de jogo
    Se(Vida do jogador inferior a 1){
        Encerra jogo por função de morte;
    }
}
```

4. Resultados e Discussão

O resultado foi bastante próximo do desejado. Embora haja um atraso na detecção natural de qualquer WebCam, o escudo é possivel jogar o jogo mesmo este sendo rapido. A detecção do escudo ficou extremamente precisa, com uma detecção rapida que permite movimentações rapidas. A detecção da espada, similarmente possui uma detecção rapida e precisa, embora devido a nosso hardware, o usuario deva tomar mais cuidado com apontar o led frontalmente. Inicialmente, foi escolhido este projeto, não apenas pela detecção de objetos diversos, mas pela construção de um jogo elaborado e bem construido, algo que foi atingido. O unico fator não atingido nos resultados esperados foi o hitbox do jogador, gerado pela subtração de fundo. A subtração de fundo do corpo do jogador deveria ser a unica região onde ele fosse acertado, permitindo esquivas no jogo. Entretanto, subtração de fundo não é um método perfeito, deixando vãos e ”Buracos”no hitbox do jogador, especialmente em regiões onde a cor do jogador e do fundo são bem similares. Mesmo assim, manteve-se a subtração de fundo, não como hitbox, mas apenas como representação grafica do jogador no jogo.

5. Conclusão

Chegamos onde queriamos. Isto pode ser dito com certeza. O conhecimento prévio de allegro foi relevante, entretanto nós dois trabalhamos no BEPiD, e lá realizamos varios pequenos projetos de lógica similar, que nos permitiu realizar este projeto maior facilidade. A mesma lógica usada gerou os ciclos de ataque e monstros. O grande desafio foi o reconhecimento da Camera, e concluimos que a camera pode ser um instrumento extremamente eficiente, apesar de custoso para a memória. Foi interessante também aprender a composição visual do computador e da WebCam, lições que definitivamente serão uteis no futuro.

Referências

[1] Touhou 6 - Enbodiment of Scarlet Devil

[2] Dark Souls, Atlus

[3] Multiplos, J.R.R.Tolkien

[4] Heredian, PI6 de 2013 do BCC-Senac https://github.com/celsovlpss/BCC-2s13-PI6-Heredian