Visão geral do projeto VERSUS na disciplina Projeto Integrador III

Helder Doutel e Rodrigo Melo

02 de Junho de 2015

Resumo

Projeto de jogo em 2 dimensões que visa proporcionar melhor experiência de interação entre o jogador e o jogo utilizando de visão computacional para aprimorar a interface (câmera usada para reconhecimento de padrões (cores) de forma à estipular os comandos). Durante a empreitada foi necessário desenvolver códigos que calculassem angulo e e distancia através do movimento dos pontos para que fosse possível definir os movimentos do personagem, também foram desenvolvidos algoritmos que compreendessem colisões para tornar o jogo mais dinâmico.

Palavras-chave: jogo, 2D, visão computacional, interação, tiro.

1 Introdução

O jogo foi desenvolvido de forma que ele se torne similar ao jogo Contra, sendo ele de tiro e 2D ao mesmo tempo que é possível alterar a direção do tiro. Será desenvolvido com métodos de visão computacional que irão maximizar a experiência do usuário, provendo uma interação extremamente agradável ao jogador utilizando de movimentos reais para simular ações similares no jogo e assim optimizar a usabilidade. As dificuldades na produção do jogo envolvem principalmente a captação de certos movimentos, mesmo com o auxilio da separação por cores, ainda é preciso trabalhar a lógica em cima do seu uso para poder desenvolver as métricas que serão cobradas para poder jogar o jogo com aproveitamento total. O personagem principal possui a capacidade de pular para desviar dos projéteis inimigos, de alterar o angulo do seu tiro e inclusive de anular os projéteis inimigos com os seus próprios colidindo-os no ar. Os tiros inimigos virão de pontos arbitrários do lado da tela deixando o jogo mais dinâmico e menos previsível. Os inimigos terão como função tentar passar pelo personagem para invadir a sua base, a partida termina quando 5 inimigos passarem pelo personagem ou quando o personagem for alvejado 5 vezes.

2 Revisão da literatura

Ao ponderar o tema do jogo, primeiro foi decidido o gênero como de tiro, e tendo em mente que o jogo seria em 2D, as principais inspirações foram os jogos Contra e Metal Slug pois são jogos que demonstram dinamismo e interface amigável ao usuário (mesmo que com os controles habituais), sendo que já é possível pular nesses jogos, e também é possível alterar a direção do tiro. Em ambos é preciso tomar cuidado para não ser atingido.

3 Desenvolvimento

Para produzir a interação do jogador com o jogo em si, será preciso utilizar uma arma fictícia que será reconhecida por uma câmera que estará posicionada ao lado do jogador, podendo assim identificar a altura da arma, movimento de pulo do jogador dentre outros. O código será escrito em C, e usará como apoio Allegro 5 em conjunto com métodos de visão computacional (OpenCV) para identificação de padrões e etc via câmera. Serão usados códigos para reconhecimento de cores via camera para poder definir o resultado das ações do jogador com o mundo virtual, basicamente, o ângulo da arma que ele empunhará, também está sendo pesquisada a melhor forma de detectar o pulo no jogo, as ferramentas utilizadas na parte programacional serão as padrão e o resto do código seguirá de forma usual para criação de um jogo e de sua mecânica (contadores e etc). Durante a criação do jogo será preciso usar a densidade das cores, seleção de pontos, detecção de múltiplas cores assim como a intensificação de cores, para ter mais precisão e assertividade nos dados que serão manipulados pelo código.
Detecção de dois pontos:
Para detectar um ponto específico na imagem a função primeiramente varre a matriz, ressalta a cor de interesse, faz uma nova varredura da matriz com os novos valores. A partir desse ponto a matriz trabalha com o conceito de densidade de cor, assim desconsiderando pixeis perdidos na imagem, para cada pixel da cor desejada, os pixeis em volta são comparados e se todos forem da mesma cor, a ponto é selecionado, se o ponto estiver mais ao centro da imagem ele é substituído (Figura 1).
O processo ocorre duas vezes, já que estamos detectando duas cores. Utilizando os dois pontos escolhidos, uma nova função faz o calculo da distância entre dois pontos, está distancia é a hipotenusa(h), com isso temos as 3 distâncias do triângulo retângulo, xb-xa, yb-ya e h, assim, calculamos o seno (cateto oposto/hipotenusa) do ângulo formado. Podendo assim controlar inclinação da arma em 3 níveis (Figura 2).

Detecção de pulo:
Utilizando um dos pontos selecionados a função compara o ponto anterior com o ponto atual, o ponto anterior é substituido no final da função, a função roda a cada 5 frames do jogo, se

a variação entre o ponto for muito grande entre um ciclo e outro é detectado o pulo (Figura 3).
O jogo funciona com base nessas 3 funções, os resultados são utilizados para disparar contadores e alterar variáveis.

4 Considerações Finais

A idéia é que o jogo seja de tiro contínuo, assim, o jogador só precisará se preocupar com as outras ações que envolvem estritamente o personagem e o jogo será para apenas um jogador. O resultado será um jogo dinâmico e agradável para que não se torne incômodo de controlar com as ações, ou até mesmo cansativo. Toda a experiência do usuário será levada em consideracão para poder definir as melhores formas de desenvolver o material proposto. Durante o desenvolvimento, as maiores dificuldades foram devido à matematica aplicada as funções pois além da própria precisão matemática, foi preciso ajustá-la para que ficasse em uníssono com os algoritmos, provendo assim a melhor qualidade possível no processamento dos códigos. Conforme o trabalho se desenrolou, as idéias evoluíram e foram tomando uma forma mais amigável no ponto de vista programacional, estruturando todas as funções e necessidades básicas de um jogo de forma acessível para o usuário. Os algoritmos gerados possuem uma ótima precisão de reconhecimento tanto quanto de uso das funções, tornando o jogo dinâmico e resultando em uma interface amigável. Algumas peculiaridades podem ser observadas quando se considera o habitual, porém o próprio Allegro5 possui caracteristicas intrinsecas as quais foi necesário se adaptar. Todo o desenvolvimento foi pensado de forma que não deixasse o jogo muito pesado mas que por outro lado, o jogador tivesse acesso à acessar comandos de forma amigável e não vetada.

5 Bibliografia

http://opencv.org/documentation.html
http://www.rafaeltoledo.net/tutoriais-allegro-5/
http://1.bp.blogspot.com/-0RYv4n8fDrg/UryscS6P5-I/AAAAAAAABBE/cJh3xvHyHJo/s1600/metal-slug-sprites-80.jpg
http://pt.wikipedia.org/wiki/Contra$_{\%}$28$jogo_eletr\%C3\%B4nico\%29$
$http://animationfight.blogspot.com.br/p/sprites.html$
$http://pt.wikipedia.org/wiki/Metal_Slug_{\%}28s\%C3\%A9rie\%29$

6 Códigos

```
void pver(unsigned char ***matriz, int altura, int largura, int *rx, int *ry){
    int baixo, direita, esquerda, cima;
    int r, g, b;
    int achoux, achouy;
    *ry = 0;
    for(int x = 0; x < altura ; x++){
        for(int y = 0; y < largura ; y++){
            r = matriz[x][y][0];
            g = matriz[x][y][1];
            b = matriz[x][y][2];
            if(r == 255 && b == 0 && g == 0){
                if(x + 5 < altura && x - 5 > 0){
                    cima = matriz[(x + 5)][y][0];
                    baixo = matriz[(x - 5)][y][0];
                }
                if(y + 5 < largura && y - 5 > 0){
                    direita = matriz[x][(y + 5)][0];
                    esquerda = matriz[x][(y - 5)][0];
                }
                if(cima == 255 && baixo == 255 && direita == 255 && esquerda == 255 && y > *ry){
                    *rx = x;
                    *ry = y;
                }
            }
            cima = 0;
            baixo = 0;
            direita = 0;
            esquerda = 0;
        }
    }
    return;
}
```

Figura 1 – Dois pontos

```
int ang (int rx, int ry, int bx, int by){
    int xa = 0, xb = 0, ya = 0 , yb = 0;
    int dx, dy;
    float d;
    float seno;
    int angulo = 0;

    if(ry < by){
        xa = ry;
        xb = by;
    }
    else{
        xb = ry;
        xa = by;
    }
    if(rx < bx){
        ya = rx;
        yb = bx;
    }
    else{
        yb = rx;
        ya = bx;
    }

    if(xa == 0 || xb == 0 || ya == 0 || yb == 0)
        return 0;

    //printf("xa = %d, xb = %d, ya = %d, yb = %d\n", xa, xb, ya, yb);
    dx = xb - xa;
    dy = yb - ya;
    d = (dx*dx) + (dy*dy);
    //printf("d %f\n", d);
    d = sqrt(d);
    //printf("raiz %f\n", d);
    //printf("dx %d dy %d\n", dx, dy);
    seno = dy/d;

    //printf("%f\n", seno);
    if (seno > 0.34 && seno < 1 && bx < rx){
        //printf("45\n");
        angulo = 1;
    }
    if (seno > 0.34 && seno < 1 && bx > rx){
        //printf("- 45\n");
        angulo = -1;
    }

    return angulo;

}
```

Figura 2 – Angulo

```
int pega(int rx, int ry, int bx, int by, int *srx, int *sry, int *sbx, int *sby){
    //printf("%d %d\n", rx, *srx);
    int x = 0;
    if(rx < *srx - 50 && rx != *srx){
        //printf("pulou: %d %d\n", rx, *srx);
        x = 1;
    }

    *srx = rx;
    *sry = ry;
    *sbx = bx;
    *sby = by;
    return x;
}
```

Figura 3 – Pulo