Departamento de Eng. Eletrotécnica e de Computadores

FCTUC

Projecto Final

Projecto de Sistemas Digitais

Professor: Jorge Lobo

Shuri-link

Autores:

João Ferreira 2013139657

Email: uc2013139657@student.uc.pt

José Castanheira 2013139490

Email: uc2013139490@student.uc.pt

Data:

07/07/2017

Descrição sumaria

Para o nosso projecto final decidimos criar um jogo tendo por base as mecânicas de um jogo da nossa infância "Bowman" e a arte do jogo "Legend of zelda".

No nosso jogo que, apelidamos de "Shuri-link", o jogador tem como objectivo acertar do adversário com uma shuriken podendo escolher a força inicial com que esta é lançada. Ao iniciar o jogo, depois de sair do main menu, o utilizador tem a sua personagem com uma barra atrás que representa a força. A força está sempre a aumentar até atingir um máximo e nesse caso volta a zero. Apos clicar no botão esquerdo do rato, ou no touchsreen, é usado o valor da força seleciona para lançar a shuriken. Caso esta acerte no alvo o utilizador recebe um ponto e o alvo aparece numa nova posição aleatória. Quando falha aparece o menu de game over e volta-se para o menu principal.

Inputs existentes

Para interagir com o jogo o utilizador pode usar os seguintes inputs, botão direito do rato quando está no main menu iniciar o jogo, botão esquerdo do rato para selecionar a força, touchscreen para fazer ambos. A key[1] reinicia o jogo no estado inicial do main menu e a key[2] vai para o estado de Game_over/Stop.

Material utilizado

Para realizar o projecto usamos a placa FPGA Altera DE2 Cyclone II, o modulo de touchpad TRDB LTM, um rato com conexão ps/2 e o monitor VGA. Com este material conseguimos por em prática tudo o que aprendemos al longo do ano.

Overview geral da implementação

A implementação foi feita maioritariamente em VHDL, por estarmos mais familiarizados e pode ser dividida em 5 partes, seleção da força e coordenadas actuais da shuriken, geração e coordenadas actuais do Dark Link(o alvo), verificador de colisões , display da cena e máquina de estados.

A geração da força é feita com um contador de 1 até 10 e quando o utilizador carrega no touch Screen ou no botão direito do rato este valor é guardado e passamos ao lançamento da shuriken. A shuriken tem associadas as equações de movimento, que com a força selecionada são responsáveis pelo movimento parabólico da shuriken.

As coordenadas do Dark Link são obtidas usando o valor de um contador de 0 a 15 para simular uma posição aleatória. Quando o jogador acerta no dark link, é guardado o valor que esta no contador e este é somado a uma posição inicial por nós definida. Assim, a primeira

posição do alvo é sempre a mesma, mas a partir do momento que se acerta passa a ser aleatória.

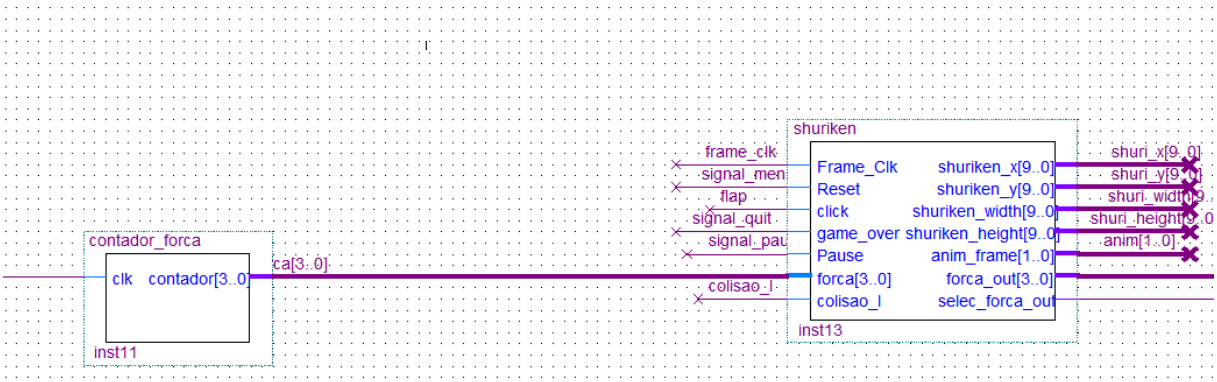
Para saber se existe colisões com o chão ou com o dark link foi feito um bloco que a partir das posições verifica as condições de colisão. Caso exista colisão com o dark link é activada uma saída que irá permitir que incrementar o contador do score em '1' unidade. Caso exista uma colisão com o chão o utilizador perde e é activada a saída que é responsável pelo jogo passar para o menu de game over.

O display da cena faz uso das coordenadas dinâmicas das shuriken , do dark link e da força para ir actualizando as suas sprites no monitor. Os outros elementos estáticos são desenhados conforme o estado em que nos encontramos. Aqui também foi feito as codificações das sprites.

Construímos uma máquina de estados que comanda o estado do jogo que estamos, main menu , a jogar ou game over. A maquina de estados tem como entradas o bit das colisões com o chão, as KEY[2] e KEY[1], se existiu algum clique esquerdo ou toque no touch screen.

Implementação detalhada

Shuriken



Na imagem temos o bloco por nós criado para tratar da posição da shuriken e da seleção da força. Este tem como entradas o estado em que nos encontramos dado pelos signals para agir conforme o necessario, o clock do VGA pois as atualizações não precisam de ser maiores que o clock do monitor. A entrada click , que nos diz se o botão esquerdo do rato ou o touch screen foram premidos. A entrada forca [3..0] é um valor sempre a alterar do contador incremental da forca. A entrada colisão_l diz-nos se acertamos no nosso alvo ou não.

Dentro do bloco shuriken existem dois momentos diferentes que são a seleção da força e a movimentação do shuriken.

A seleção da força é feita sempre que se entra no estado, a jogar ou após uma colisão com o alvo. A entrada forca[3..0] vem de um contador que varia entra 0 e 10 de forma linear e quando a entrada click vai a um é copiado o valor da forca de entrada para uma variável chamada forcaR usada nos cálculos.

Após a seleção da força é calculada a velocidade inicial y e a velocidade inicial x pelo seguinte expressão:

Velocidade inicial de y= -18 – forcar
Velocidade inicial de x= 5 + forcaR

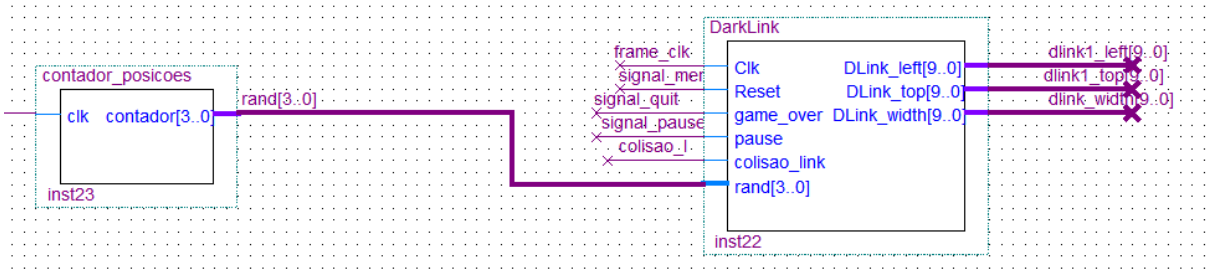
A velocidade de y é negativa pois o as coordenados do monitor do eixo do y são 0 no topo. Depois de termos os valores iniciais das velocidades estas são somadas à posição x e y da shuriken. A velocidade de x é constante ao longo do funcionamento do programa mas a velocidade de y vai variando:

Velocidade de y = velocidade de y + gravidade

A gravidade é constante ao longo do funcionamento e após testes chegamos à conclusão que o melhor valor era 1.

O bloco tem como saídas a posição x e y da shuriken, a sua largura e altura que são precisas para o calculo das colisões e para o desenho no ecrã. A força que foi escolhida e se estamos se a escolhemos ou não, isto é usado para fazer display no visor de 7 segmentos da força escolhida. Apesar de termos uma saída 'anim_frame' que escolheria que sprite da shuriken usar, após testes chegamos à conclusão que não gostávamos da sprite da rotação da shuriken pelo que a deixamos de usar.

Dark Link

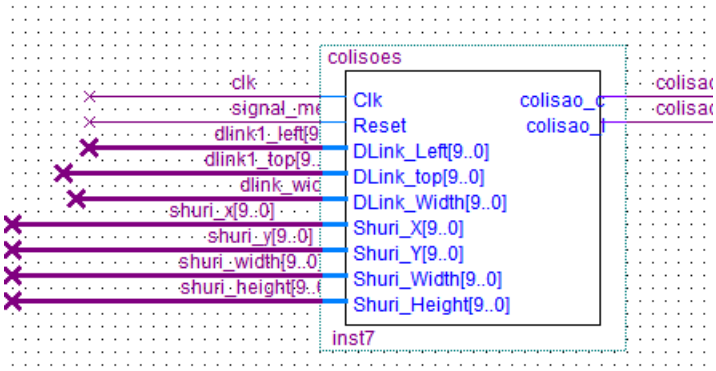


Os blocos a cima representados são responsáveis pela escolha das coordenadas para o dark link. O bloco tem como entradas o frame clock pois os cálculos não precisam de ser mais rápidos que o clock do monitor, os bits dos estados dados pelos signal para se saber o estado que nos encontramos. A entrada colisão_l que diz se houve colisão com o dark link ou não, e a entrada rank [3..0] é o valor que vai ser somado à posição inicial para esta ir alterando ao longo do tempo.

A primeira posição é sempre a mesma mas a partir dai é sempre somado o valor aleatório quando há colisão para o novo dark link aparecer num novo sitio.

O bloco tem como saída as coordenadas x e y do dark link assim como a largura do mesmo.

Colisões



O bloco das colisões tem como entradas as posições x e y quer do dark link quer da shuriken assim como a altura e largura da shuriken e a largura do dark link.

O funcionamento é bastante simples e verfica se as coordenas de ambos se intersectam. Também é feita a verificação de colisão com o chão, ligeiramente abaixo do sition onde estão as personagens. Não foi feita a colisão nem com o tecto máximo pois pelas equações da velocidade este é impossível de alcançar.

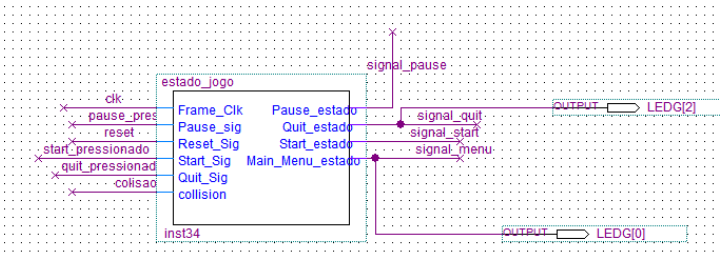
A saída colisão vai a 1 quando existe colisão com o chão, e leva ao game over.

A saída colisão_l vai a 1 quando se acerta no dark link e leva a um incremento do score e a ao resto das coisas faladas nos outros blocos.

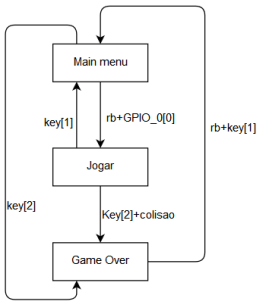
VGA

Foi implementado o bloco fornecido pelo professor nas aulas práticas mas alteramos o numero de bits de cada cor de forma a ter uma maior gama de cores

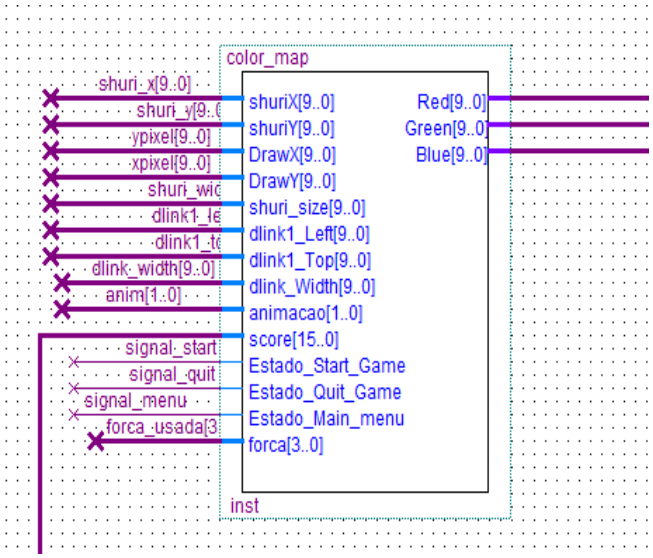
Maquinas de estados



Implementamos uma máquina de estados com 3 estados em que tem como objectivo distinguir as 3 cenas existentes sendo elas , a cena de main menu, a cena a jogar e a cena de game over.



Color map



Este bloco é a nossa forma de dizer que cor o bloco VGA_SYNC vai ter de "pintar" em determinado píxel.

São passados como inputs as posições de todas as nossas sprites dinâmicas e os estados do jogo (para apenas desenhar certas coisas no devido estado).

A partir das posições das sprites, o bloco vai dar o valor da cor do píxel respetivo.

Estas sprites estão guardadas e codificadas neste bloco em arrays que declaramos no início do bloco. Estas arrays são percorridas conforme bits como "shuri_on" passem a 1 (drawX encontra-se sobre a posição que o shuriken deve estar), e o bloco tem como output as cores respectivas, que codificámos no fim.

```
if((DrawX >= CONV_STD_LOGIC_VECTOR(15, 10) and DrawX <= CONV_STD_LOGIC_VECTOR(18, 10) and
DrawY >= CONV_STD_LOGIC_VECTOR(280, 10) and DrawY <= CONV_STD_LOGIC_VECTOR(360, 10)) or --barra esquerda
(DrawX >= CONV_STD_LOGIC_VECTOR(15, 10) and DrawX <= CONV_STD_LOGIC_VECTOR(40, 10) and
DrawY >= CONV_STD_LOGIC_VECTOR(280, 10) and DrawY <= CONV_STD_LOGIC_VECTOR(283, 10)) or -- barra de cima
(DrawX >= CONV_STD_LOGIC_VECTOR(15, 10) and DrawX <= CONV_STD_LOGIC_VECTOR(40, 10) and
DrawY >= CONV_STD_LOGIC_VECTOR(357, 10) and DrawY <= CONV_STD_LOGIC_VECTOR(360, 10)) or --barra de baixo
(DrawX >= CONV_STD_LOGIC_VECTOR(37, 10) and DrawX <= CONV_STD_LOGIC_VECTOR(40, 10) and
DrawY >= CONV_STD_LOGIC_VECTOR(280, 10) and DrawY <= CONV_STD_LOGIC_VECTOR(360, 10))) then
    color:= 4;
end if;
```

O código VHDL da figura acima é um exemplo da maneira como desenhamos uma barra branca, atrás do Link normal. Comparamos a coordenada X onde o monitor atual esteja e dizemos que se esta estiver entre intervalos desejados, a cor do píxel nesse sítio passa a ser 4, que codificámos no fundo do bloco como a cor branca(R,G,B= [1024,1024,1024].

NOTA: Uma parte deste bloco foi baseada numa implementação de um grupo do ano passado(do Jorge Monteiro e Alexandre Martins), conforme foi dito ao professor no LAB durante a realização do projecto.