Unicorn Explosion-The Game

Designed by Andrew Swanson

[akswanson@cpp.edu](mailto:akswanson@cpp.edu)

Documented By Changcan(Benjamin) Hu

[changcanhu@cpp.edu](mailto:changcanhu@cpp.edu)

version 1.0, 2018-11-25

## Table of Contents

## Implementation

## Purpose

## Logic

Implementation

Purpose:

The score engine is meant to count the output from the Physic engine as player starts playing the game. At the end of the game finished, score\_engine will pass the register value to the display engine to display the final score of the player for the round.

Logic

* Inputs:
  + Score\_Engine shares the same clock as the entire system, therefore, once any changes of the game happen, the score will act correspondingly.
  + Score\_Engine operates base on the outputs of the Physics\_Engine, which is called score\_in.
  + Difficulty array is taken to evaluate the score for each positive score\_in input, the higher difficulties input represents, the higher score of the player receives.
  + Every operation will only start once start signal is positive.
* Output:
  + Once the game is terminated, the engine will deposit the final score and output into the 32-bit array to the display engine in order to display the final score of the player.
* Loops
  + The main logic will only be scanned and tested once clock hits its positive edge.
  + If the game is not dead, the main logic will start counting base on the input of score\_in.
  + The increment of score is based on the ratio we given by its difficulties.

|  |
| --- |
| `timescale 1ns / 1ps |
|  | ////////////////////////////////////////////////////////////////////////////////// |
|  | // Company: |
|  | // Engineer: |
|  | // |
|  | // Create Date: 11/27/2018 10:33:04 AM |
|  | // Design Name: |
|  | // Module Name: score\_engine |
|  | // Project Name: |
|  | // Target Devices: |
|  | // Tool Versions: |
|  | // Description: |
|  | // |
|  | // Dependencies: |
|  | // |
|  | // Revision: |
|  | // Revision 0.01 - File Created |
|  | // Additional Comments: |
|  | // |
|  | ////////////////////////////////////////////////////////////////////////////////// |
|  |  |
|  | //The module that tracks the player's current score. |
|  | //The player is awarded one point for every space they travel, and ten point for each block they successfully jump over |
|  | module score\_engine( |
|  | input clock\_div, |
|  | input score\_in, //score\_in is sent from the collision detection when the player jumps over a block |
|  | input [1:0]difficulty, //To add appeal to playing on higher difficulty levels, the score is multiplied by the difficulty. / |
|  | output reg [31:0]score, |
|  | input start, |
|  | input isdead |
|  | ); |
|  |  |
|  | always@(posedge clock\_div) |
|  | begin |
|  | if(~isdead) |
|  | begin |
|  | if(~start) |
|  | score = 0; //The score resets to zero on reset |
|  | if(score\_in) //The difficulty must be added by one because it starts at zero, and multiplying the score by zero would not add extra fun :P |
|  | score = score + ((difficulty + 1)\*10); |
|  | else |
|  | score = score + ((difficulty + 1)\*1); |
|  | end |
|  | end |
|  |  |
|  | endmodule |

|  |
| --- |
| `timescale 1ns / 1ps |
|  | ////////////////////////////////////////////////////////////////////////////////// |
|  | // Company: Unicorn Explosion |
|  | // Engineer: Benjamin(Changcan) Hu |
|  | // |
|  | // Create Date: 11/25/2018 02:57:30 PM |
|  | // Design Name: |
|  | // Module Name: Brain |
|  | // Project Name: Uniorn Explosion |
|  | // Target Devices: NEXYS4-DDR |
|  | // Tool Versions: Artix-XC7A100T-CSG324 |
|  | // Description: FPGA |
|  | // |
|  | // Dependencies: |
|  | // |
|  | // Revision: |
|  | // Revision 0.01 - File Created |
|  | // Additional Comments: |
|  | // |
|  | ////////////////////////////////////////////////////////////////////////////////// |
|  |  |
|  |  |
|  | module physics\_engine( |
|  | input clk, |
|  | input start, |
|  | input [15:0] map, |
|  | input jumpin, |
|  | output reg is\_dead, |
|  | output reg score, |
|  | output reg jumpout |
|  | ); |
|  | reg [1:0] position; |
|  | reg [1:0]Q; |
|  |  |
|  | always@(posedge clk) |
|  | begin |
|  | if(~start) |
|  | begin |
|  | is\_dead = 0; |
|  | end |
|  | else |
|  | begin |
|  | case(position) |
|  | 0: |
|  | begin |
|  | if(jumpin==1) |
|  | begin |
|  | position<=2'b10; |
|  | end |
|  | else |
|  | begin |
|  | position<=2'b00; |
|  | end |
|  | end |
|  | 1: |
|  | begin |
|  | position<=2'b00; |
|  | end |
|  | 2: |
|  | begin |
|  | position<=2'b01; |
|  | end |
|  | 3: |
|  | begin |
|  | position=2'b01; |
|  | end |
|  | endcase |
|  |  |
|  | case(map[15:14]) |
|  | 0: //No block |
|  | begin |
|  | score=0; |
|  | end |
|  | 1: //Low block |
|  | if(position > 2'b00) |
|  | begin |
|  | score = 1; |
|  | end |
|  | else |
|  | begin |
|  | is\_dead=1; |
|  | end |
|  | 2: //High block |
|  | if(position==0) |
|  | begin |
|  | score = 1; |
|  | end |
|  | else |
|  | begin |
|  | is\_dead=1; |
|  | end |
|  | 3: //Don't exist |
|  | begin |
|  | score = score; |
|  | end |
|  | endcase |
|  | end |
|  | end |
|  |  |
|  | always@(position) |
|  | begin |
|  | if(position == 0) |
|  | jumpout = 0; |
|  | else |
|  | jumpout = 1; |
|  | end |
|  |  |
|  | endmodule |