

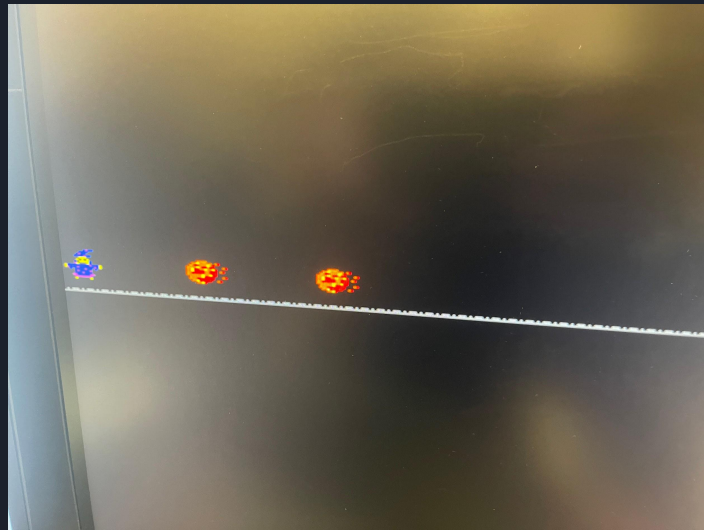
Wizard Skate



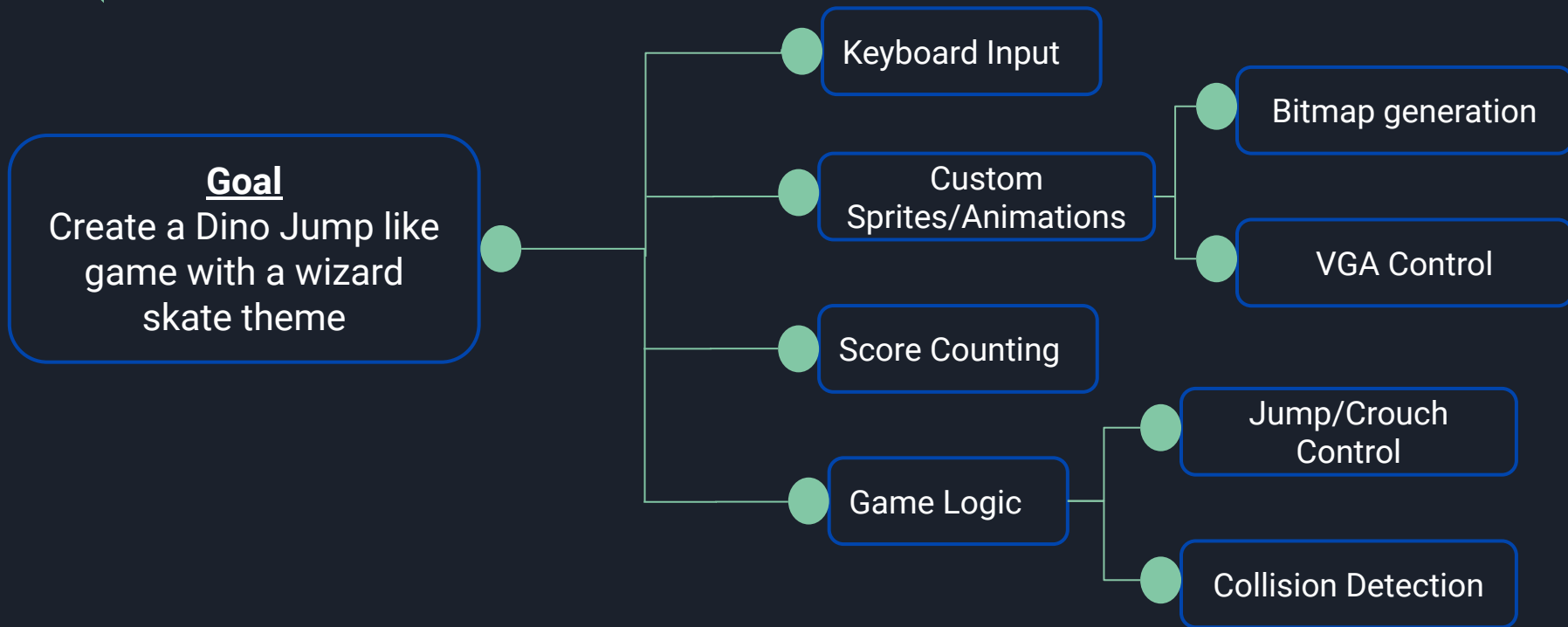
By Naeel Quasem, Luke
McCarthy, Roger Brown and
Angel Amaya

Goal/Motivation

- Create a fun, new game, similar to Dino Jump, with a new theme
- Very epic game!



Functionality

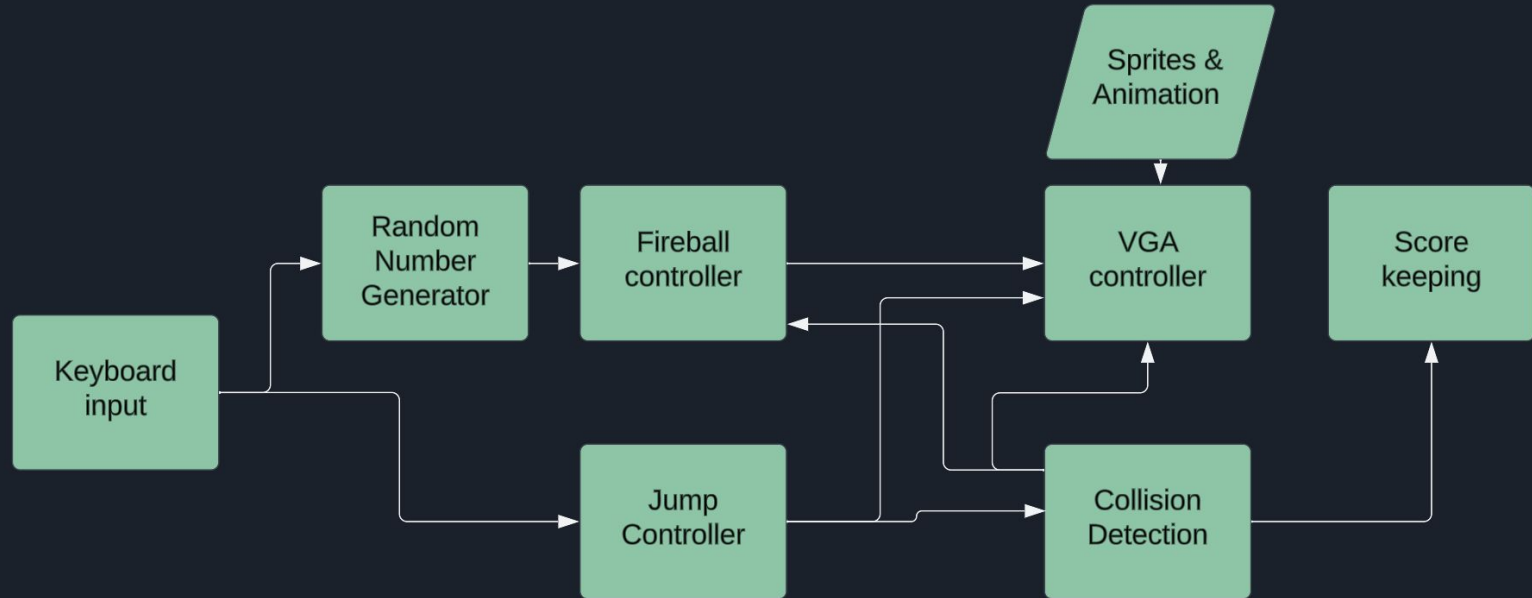




Specification

Requirements	Constraints
<ul style="list-style-type: none">• Keyboard inputs to control character• Obstacles/Collision Detection• Score-keeping• Sprites/Animations• VGA Display• Progressive Difficulty	<ul style="list-style-type: none">• Memory (for sprites)• Deadline

Block Diagram



Code Snippet: PRNG

Initialize two
8-bit registers

```
reg [7:0] seed;  
reg [7:0] hold;  
  
initial begin  
    seed = 3;  
    hold = 3;  
end
```

Increment seed
every clock cycle

```
always @(posedge clk)  
    seed = seed + 1;
```

Set hold equal to seed when
the jump button is pressed

```
always @(posedge click) begin  
    if (click == 1)  
        hold = seed;  
end
```

Multiplies hold by seed every
frame for use by other modules

```
always @(posedge frame)  
    rand = hold*seed;
```

Code Snippet

```

always @(score) begin
    for(i = 0; i <= W+(W-4)/3; i = i+1)    BCD_score[i] = 0;           // initialize with zeros
    BCD_score[-1:0] = score;                // initialize with input vector
    for(i = 0; i <= W-4; i = i+1)           // iterate on structure depth
        for(j = 0; j <= i/3; j = j+1)       // iterate on structure width
            if (BCD_score[W-i+4*j -: 4] > 4) // if > 4
                BCD_score[W-i+4*j -: 4] = b BCD_scorecd[W-i+4*j -: 4] + 4'd3; // add 3
    end
//https://github.com/AmeerAbdelhadi/Binary-to-BCD-Converter/blob/master/bin2bcd.v

```

```

always @ (posedge in_clk)begin
    count = count + 1;

    if(!collision && !reset && count%20 == 0)
        score = score+1;

    else if(reset)
        socre = 0;
end

```

```
endmodule
```

Code Snippet: Collision

```
module collision_controller(
    wyo, fxo1, fyo1, fxo2, fyo2, reset, frame, collision, crouch
);
    input reset, frame, crouch;
    input [7:0] wyo;
    input [9:0] fxo1, fyo1, fxo2, fyo2;

    output reg collision;

    always @ (posedge frame) begin
        // I'm sorry this is very cursed
        if (
            (
                (((fxo1 >= 564) && (fxo1 <= 580)) && ((0+wyo <= fyo1) && (fyo1 <= 16+wyo))) // has the first fireball hit the upright wizard ?
                || (((fxo1 >= 564) && (fxo1 <= 580)) && ((0+wyo <= fyo2) && (fyo2 <= 16+wyo))) // has the second fireball hit the upright wizard ?
                && ~crouch
            )
            ||
            (
                (((fxo1 >= 564) && (fxo1 <= 580)) && ((8+wyo <= fyo1) && (fyo1 <= 16+wyo))) // has the first fireball hit the crouching wizard ?
                || (((fxo1 >= 564) && (fxo1 <= 580)) && ((8+wyo <= fyo2) && (fyo2 <= 16+wyo))) // has the second fireball hit the crouching wizard ?
                && crouch
            )
            &&
            ~reset
        ) begin // has reset happened?
            collision = 1;
        end else begin
            collision = 0;
        end
    end
endmodule
```


Code Snippet

```
109      // wizard definition
110      if (hp < 16 && vp >= (278-wyo) && vp < (294-wyo)) begin
111          index = (vp - (278-wyo)) * 16 + hp; // Adjusted the index calculation
112          color = wizard[index]; // Access the wizard color data
113          VGA_R <= color[11:8]; // Extract the red component
114          VGA_G <= color[7:4]; // Extract the green component
115          VGA_B <= color[3:0]; // Extract the blue component
116      // fireball 1 definition
117      end else if (hp >= (620-fxo1) && hp < (636-fxo1) && vp >= (278-fyo1) && vp < (294-fyo1) && fxo1 < 641) begin
118          index_fireball = (vp - (278-fyo1)) * 16 + (hp - (620-fxo1));
119          color_fireball = fireball[index_fireball];
120          VGA_R <= color_fireball[11:8];
121          VGA_G <= color_fireball[7:4];
122          VGA_B <= color_fireball[3:0];
123      //fireball 2 definition
124      end else if (hp >= (620-fxo2) && hp < (636-fxo2) && vp >= (278-fyo2) && vp < (294-fyo2) && fxo2 < 641) begin
125          index_fireball = (vp - (278-fyo2)) * 16 + (hp - (620-fxo2));
126          color_fireball = fireball[index_fireball];
127          VGA_R <= color_fireball[11:8];
128          VGA_G <= color_fireball[7:4];
129          VGA_B <= color_fireball[3:0];
130      // line for the ground
```



Successes

- Jumping and obstacle movement work perfectly.
- Bitmap generation, sprite implementation works as expected.
- Implementation of collision.



Failures

- Code is working as expected, but sometimes has minor glitches.
- That is it.
- Thank you.