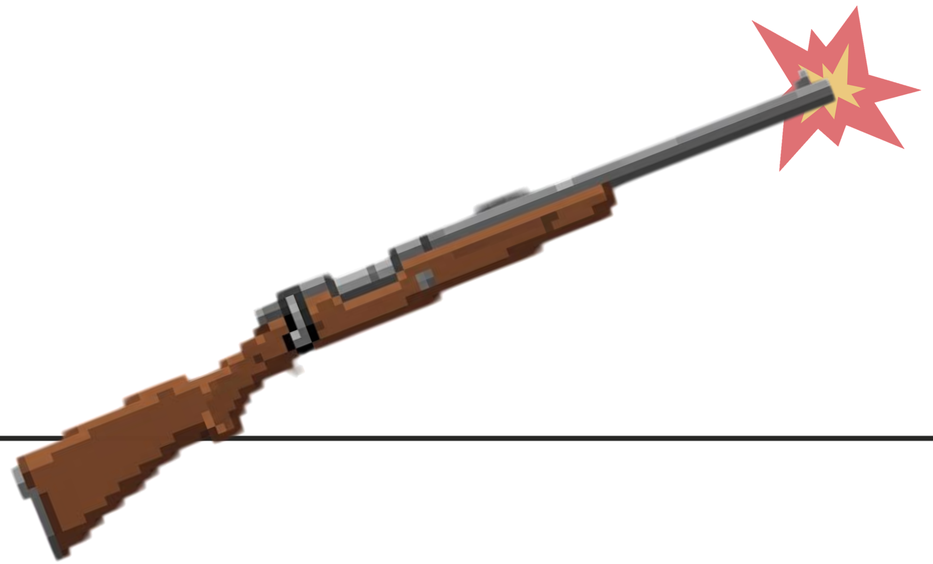
OV7670 센서를 이용한 VGA 덕헌트 게임 구현



하만 2기 12조
김문호 이재원 김태형 채준희

발표자 : 이재원

part 01

Contents

- 프로젝트 개요
- VGA Controller
- Game Controller
- 동작 영상
- Trouble Shooting
- 역할 및 고찰

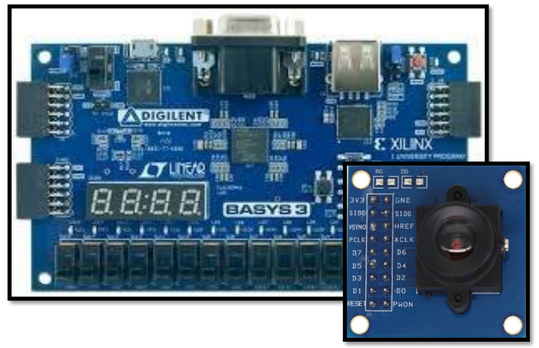
part 01

Project Overview

Design Tool: Xilinx Vivado

Language: SystemVerilog

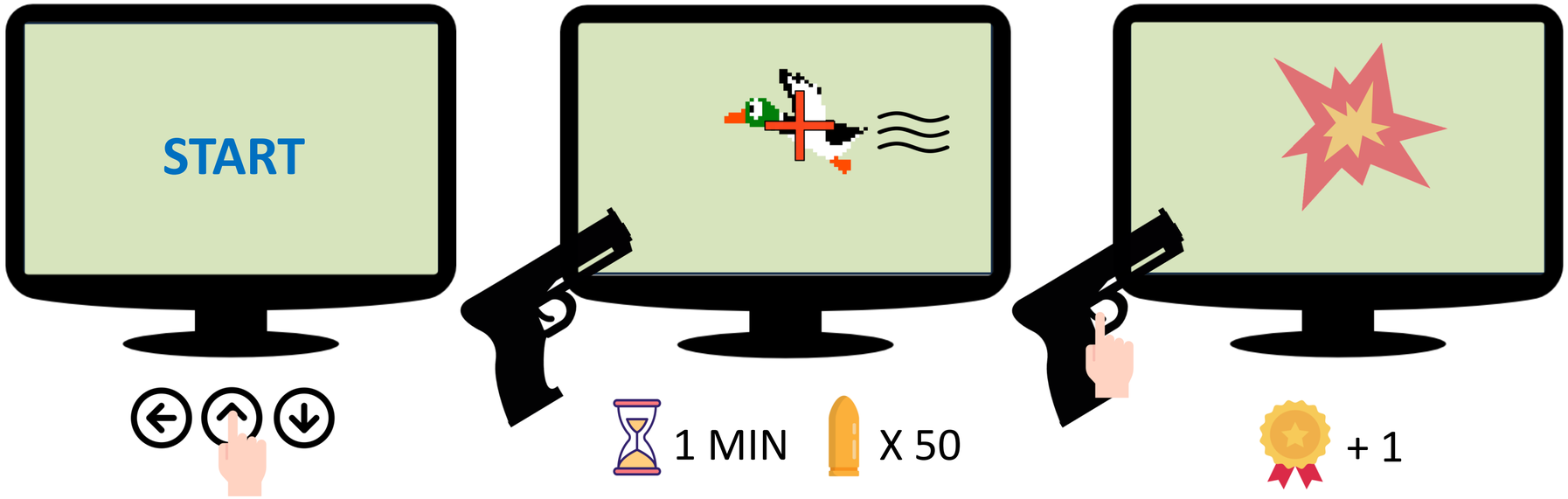
Editor: Visual Studio Code



Main Hardware Setup:
Basys3, OV7670

part 01

Project Overview

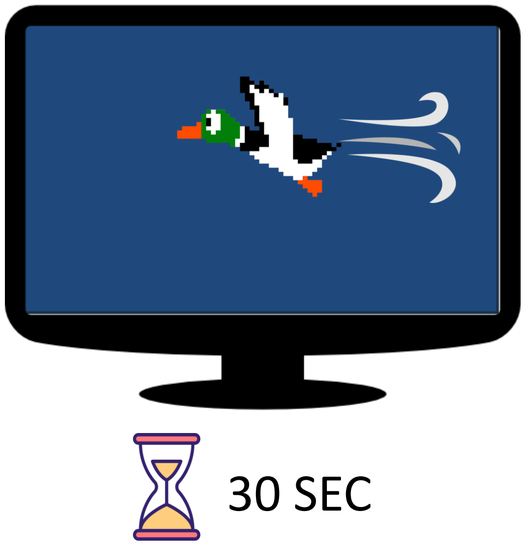


UP 버튼을 클릭해서 시작

50발의 총알로 1분동안 날아가는 오리를 조준해서 사격,
잡으면 스코어 1 증가

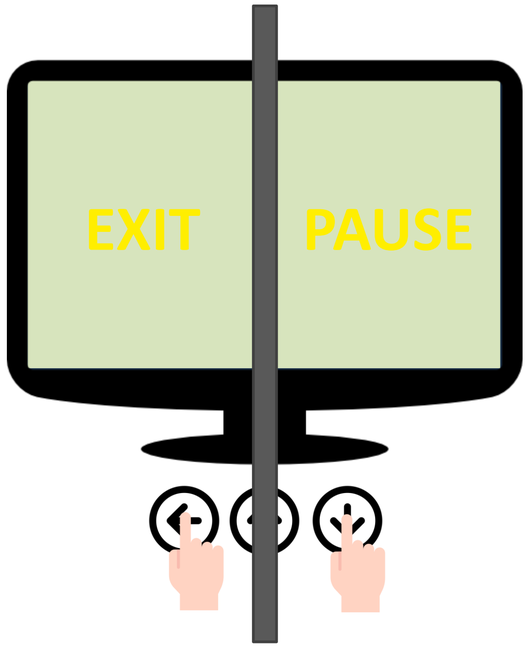
part 01

Project Overview



30 SEC

30초 남았을 때
오리가 빨라지는 야간모드 시작



LEFT 버튼으로 게임종료,
DOWN 버튼으로 일시정지

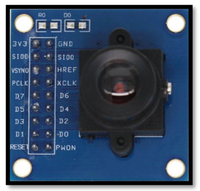
OR /

총알을 모두 사용하거나
1분이 지나면 게임 종료하고 스코어 출력

part 02

VGA Controller

OV7670 카메라로부터 받은 영상 데이터를 VGA 화면에 출력하기 위해 구현한 VGA Controller



Frame Buffer

VGA Controller

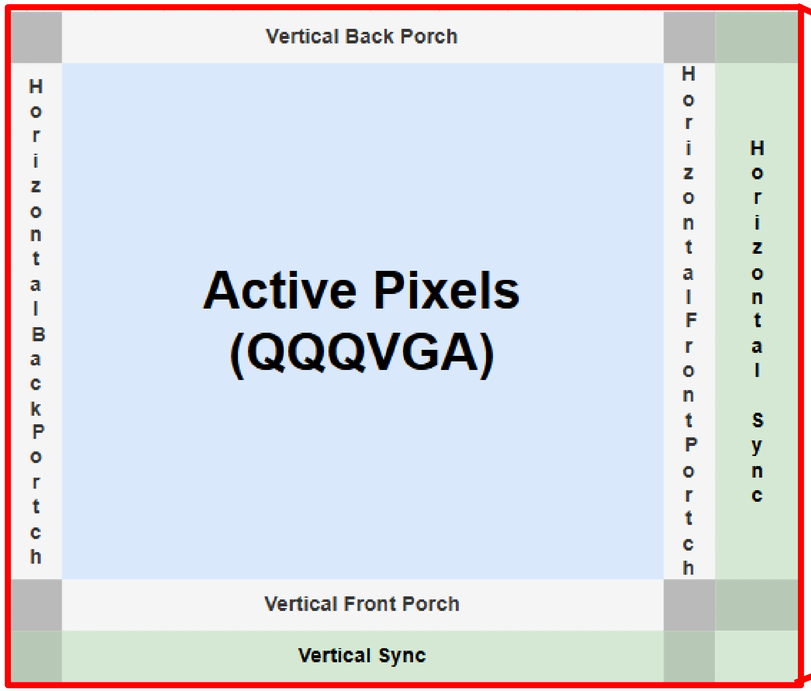
Frame Buffer에 OV7670 데이터를 저장

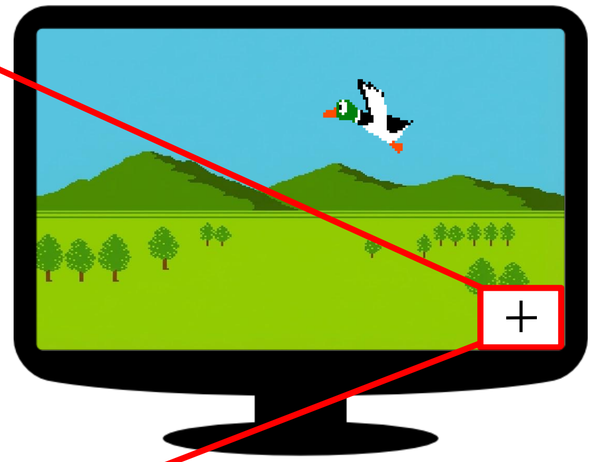
VGA Controller는
VGA 타이밍에 따라
영상 데이터와 메모리파일을 화면에 출력

part 02

VGA Controller

OV7670 카메라로부터 받은 영상 데이터를 VGA 화면에 출력하기 위해 구현한 VGA Controller



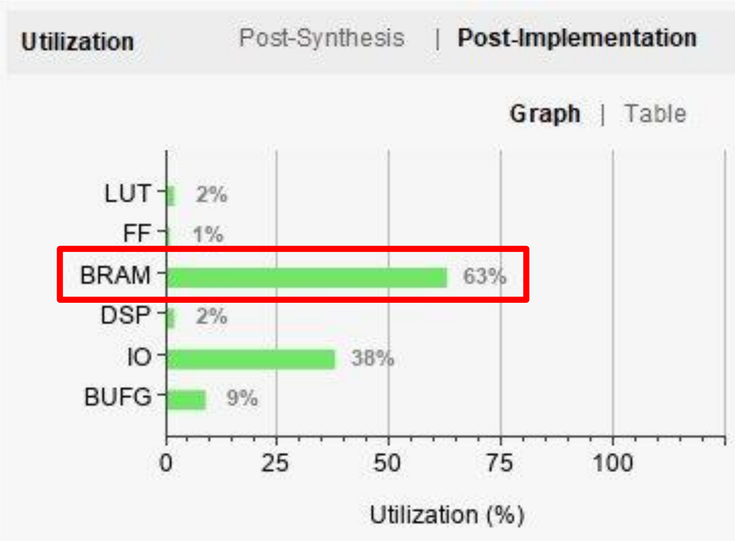


QQQVGA (80 x 60) 해상도 데이터 전송

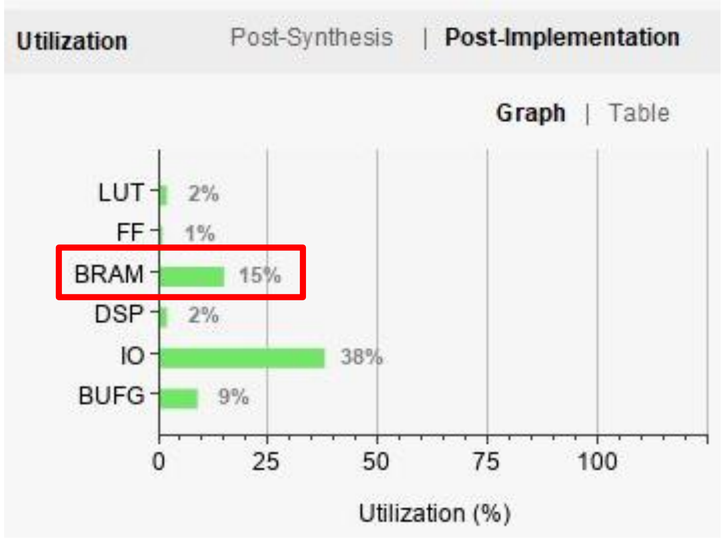
part 02

VGA Controller

영상 데이터 ROM 용량을 QVGA 대비 16배 축소, BRAM 48% 절감

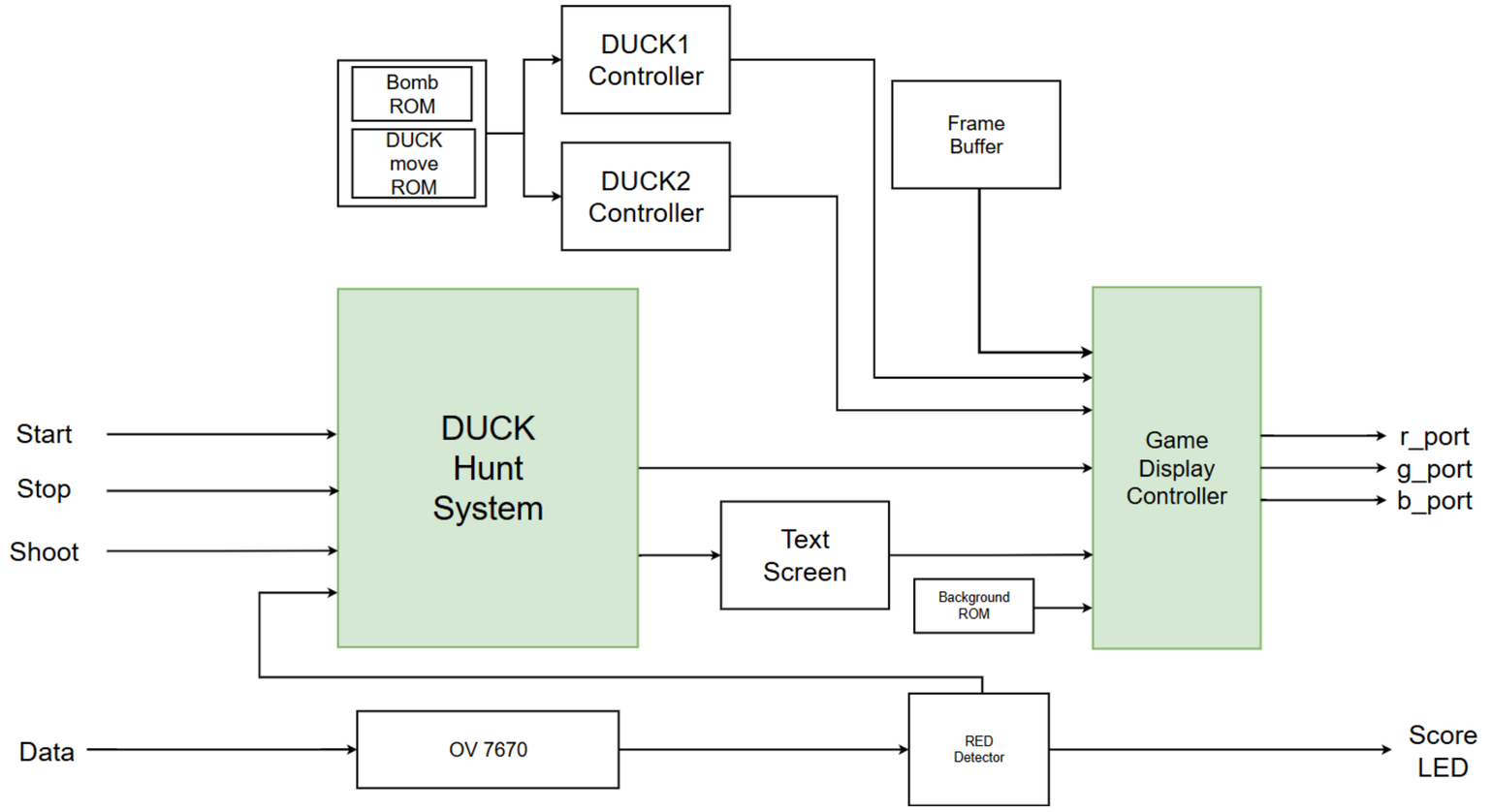


QVGA (320 x 240)



QQQVGA (80 x 60)

Block Diagram



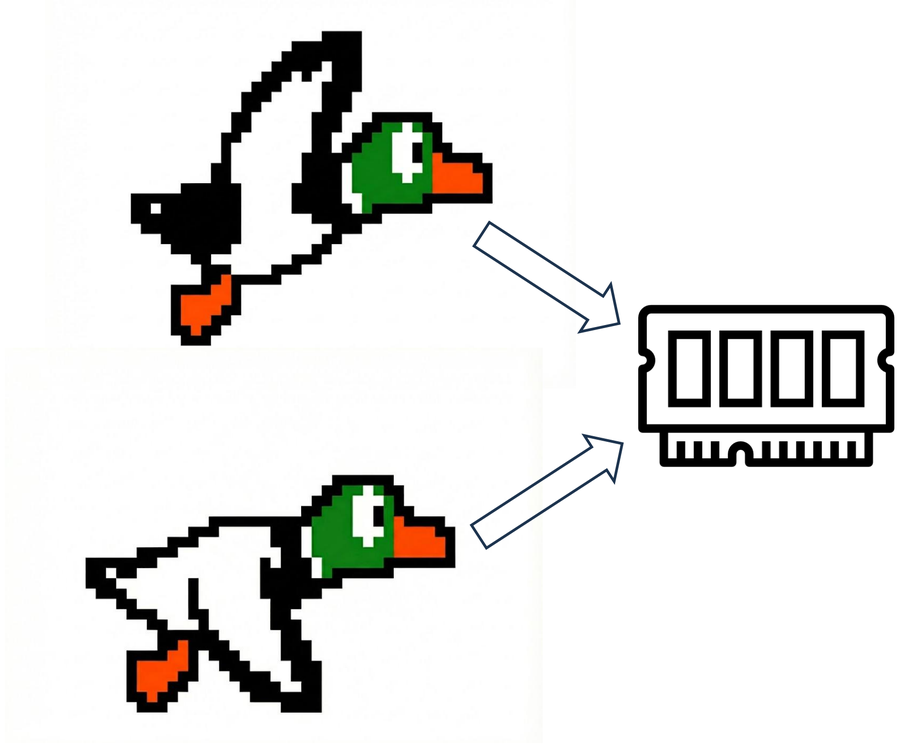
part 03

Game Controller

- 오리 움직임 구현
- 오리 포착 방법
- 야간 모드

Game Controller

오리 움직임 구현

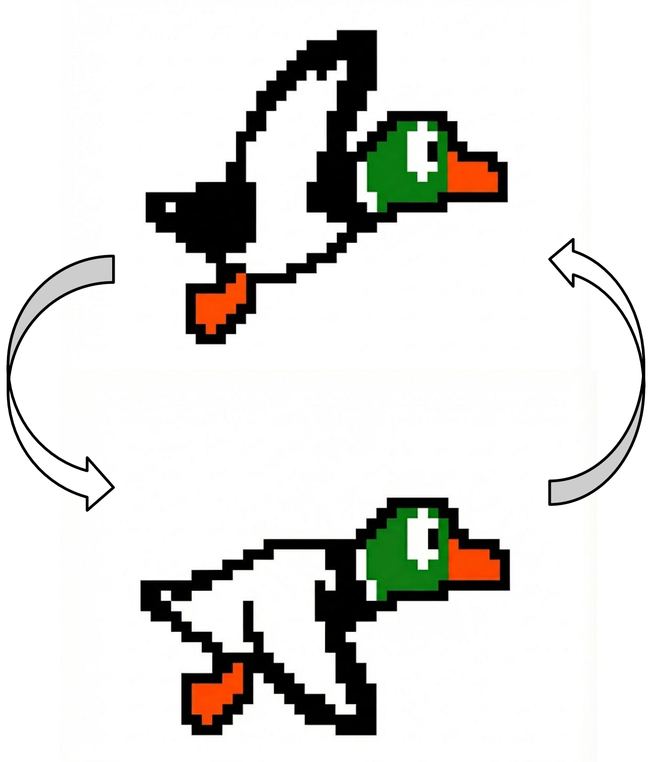


```
module Duck_Rom_Code #(
    parameter DATA_WIDTH = 16,
    parameter ADDR_WIDTH = 13,
    parameter MEM_DEPTH  = 8192,
    parameter MEM_FILE   = "duck_move.mem"
) (
    input logic clk,
    input logic [ADDR_WIDTH-1:0] addr,
    output logic [DATA_WIDTH-1:0] data
);
    logic [DATA_WIDTH-1:0] rom[0:MEM_DEPTH-1];
    initial $readmemh(MEM_FILE, rom);
    always_ff @(posedge clk) data <= rom[addr];
endmodule
```

두 가지 모션의 오리 이미지를 하나의 메모리 파일에 저장

Game Controller

오리 움직임 구현



```
if (anim_cnt < ANIM_SPEED) anim_cnt <= anim_cnt + 1;
else begin
     anim_cnt <= 0;
     frame_toggle <= ~frame_toggle;
end
```

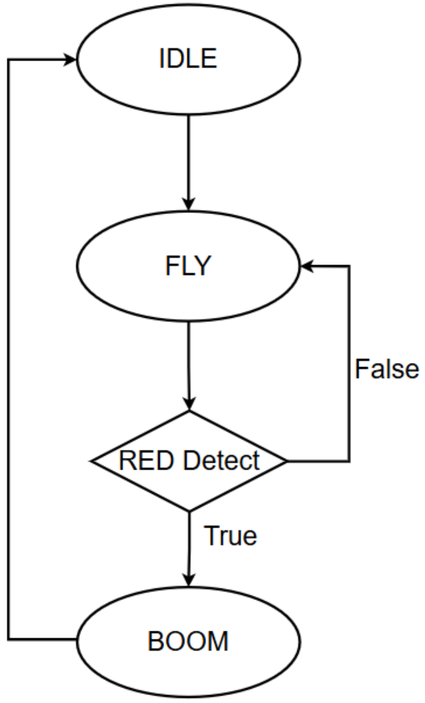
5,000,000 클럭 사이클(0.05sec) 마다 frame_toggle

```
if (state == S_FLY) begin
    base_addr = (frame_toggle) ? FRAME_OFFSET : 0;
    if (is_facing_left)
        rom_addr = base_addr + (row_idx * DUCK_W) + ((DUCK_W - 1) - col_idx);
    else rom_addr = base_addr + (row_idx * DUCK_W) + col_idx;
```

frame_toggle 값에 따라 메모리파일에서 출력하는 주소값 변경

Game Controller

오리 움직임 구현



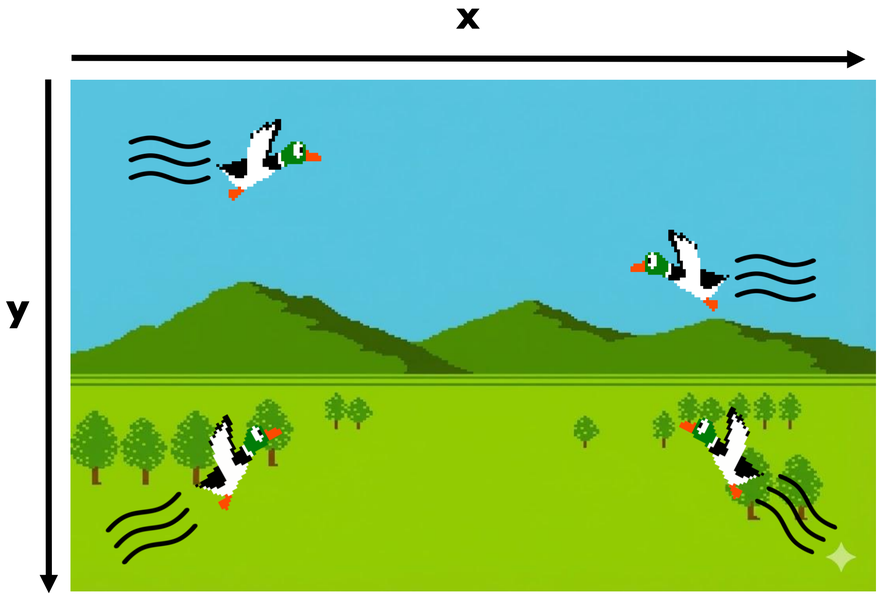

```
S_FLY: begin
case (spawn_type)
    0: begin
        if (!is_facing_left) begin
            if (duck_x < H_RES - DUCK_W)
                duck_x <= duck_x + 1;
            else is_facing_left <= 1;
        end else begin
            if (duck_x > 0) duck_x <= duck_x - 1;
            else state <= S_IDLE;
        end
    end
    1: begin
        if (duck_x > 0 && duck_y > 0) begin
            duck_x <= duck_x - 1;
            duck_y <= duck_y - 1;
        end else state <= S_IDLE;
    end
    2: begin
        if (duck_x < (H_RES - DUCK_W) && duck_y > 0) begin
            duck_x <= duck_x + 1;
            duck_y <= duck_y - 1;
        end else state <= S_IDLE;
    end
end
```

오른쪽으로 이동

왼쪽으로 이동

왼쪽으로 대각으로 이동

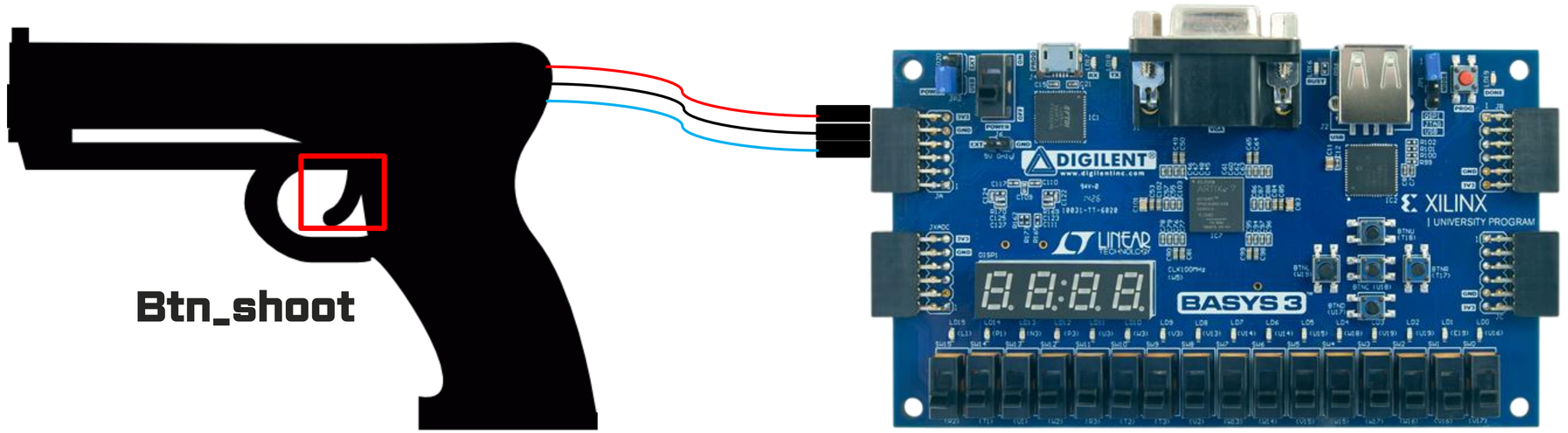
오른쪽으로 대각으로 이동



0,1,2 중 하나의 랜덤 값으로 오리가 생성되는 위치가 변하며
수평, 우측/좌측 대각선 이동 로직 랜덤 생성

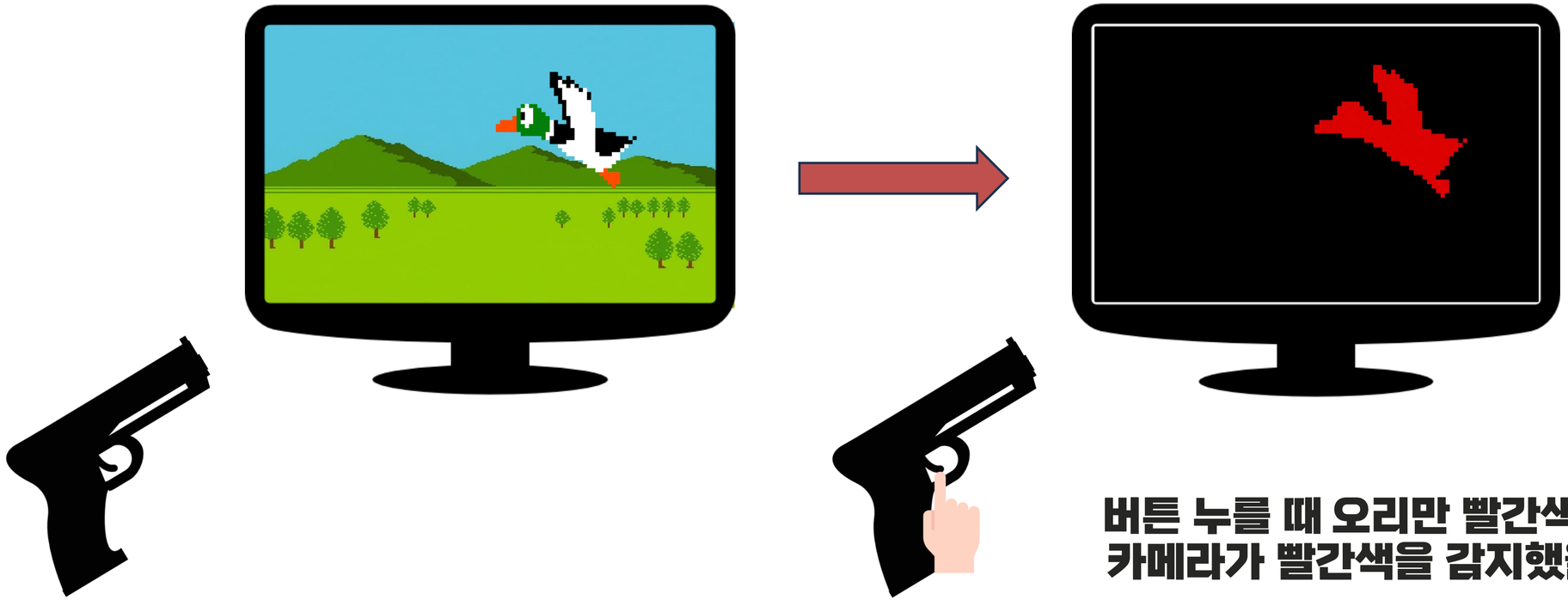
Game Controller

오리 포착 방법



Game Controller

오리 포착 방법



버튼 누를 때 오리만 빨간색으로 발광
카메라가 빨간색을 감지했을 시 명중

Game Controller

오리 포착 방법

```
if (in_detect_area && is_red_pixel) begin
    red_count <= red_count + 1;
end
```

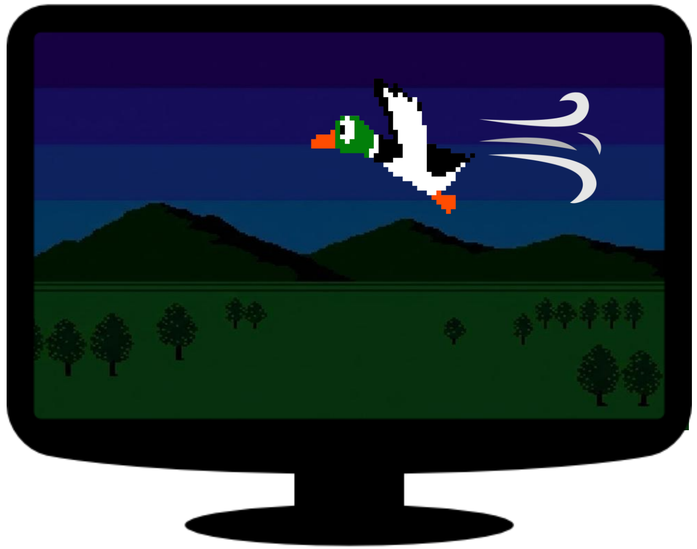
카메라의 감지영역 & 빨간색 픽셀인지 확인

```
if (!prev_vsync && vsync) begin
    frame_ended <= 1'b1;
    red_pixel_count <= red_count;
    red_detected <= (red_count >= MIN_RED_PIXELS);
```

감지 영역 내의 픽셀이 임계치 이상일 때 오리 감지

야간 모드

게임 모드 변경



```
if (invert_bg) begin
    temp_r = bg_r >> 2;
    temp_g = bg_g >> 2;
    temp_b = bg_b >> 1;
end
```

30초가 남았을 때 배경의 RGB값이 Shift되며 야간필터의 효과

```
assign current_speed_limit = (spawn_type == 0) ?
        (hurry_up ? 19'd110_000  : 19'd150_000) :
        (hurry_up ? 19'd150_000 : 19'd250_000);
```

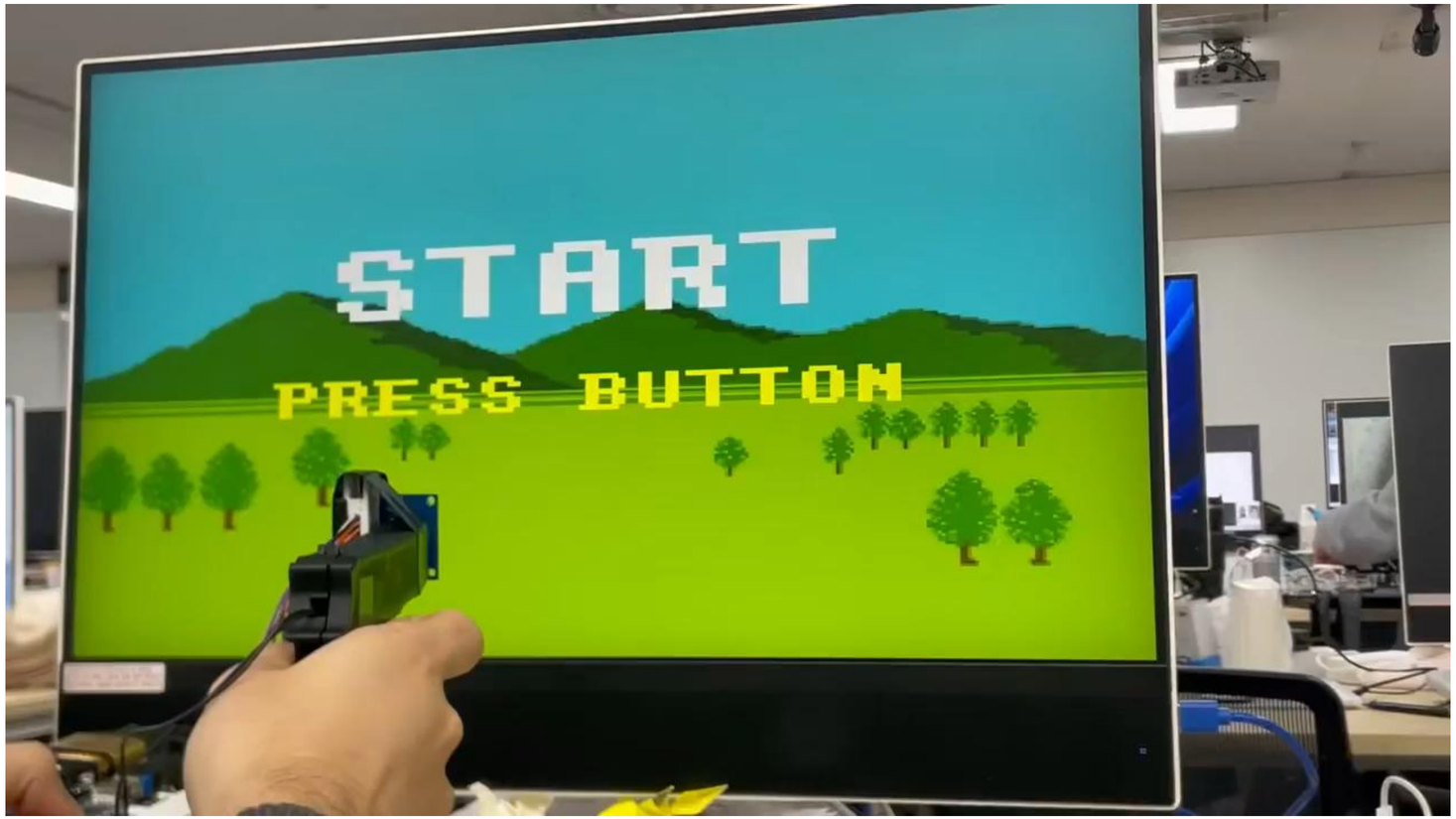
야간모드가 적용되었을 때 움직이는 오리의 속도가 증가

part 04

동작영상

part 04

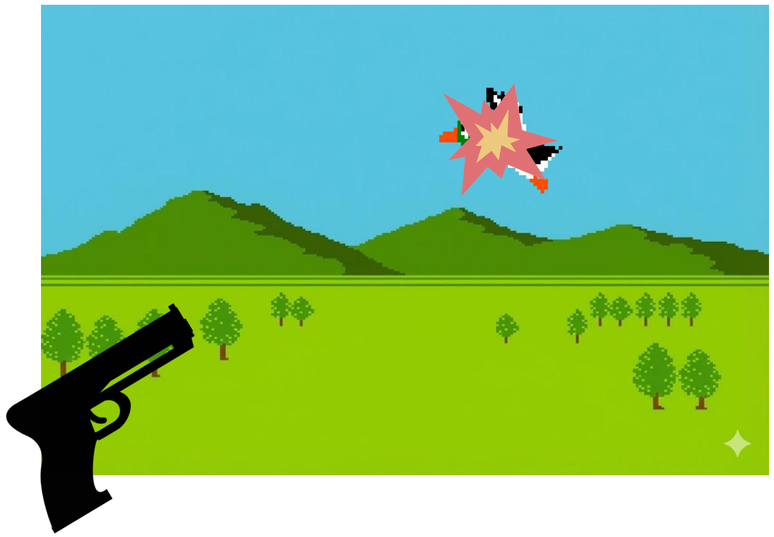
동작영상



part 05

Trouble Shooting

폭발 이펙트 간헐적 미표시 문제



- 문제 : 오리가 정상적으로 사라지고 스코어도 올라가지만 폭발 이펙트가 간혹 출력되지 않음
- 원인 : hit 신호를 놓치면 상태 전이가 되지 않아 폭발이 발생하지 않음
  → hit 신호의 지속시간 부족
- 해결방안 :

```
end else if (duck1_active && !duck1_exploding) begin
    score <= score + 1;
    hit1_pulse_cnt <= 8'd15;
end else if (duck2_active && !duck2_exploding) begin
    score <= score + 1;
    hit2_pulse_cnt <= 8'd15;
end
```

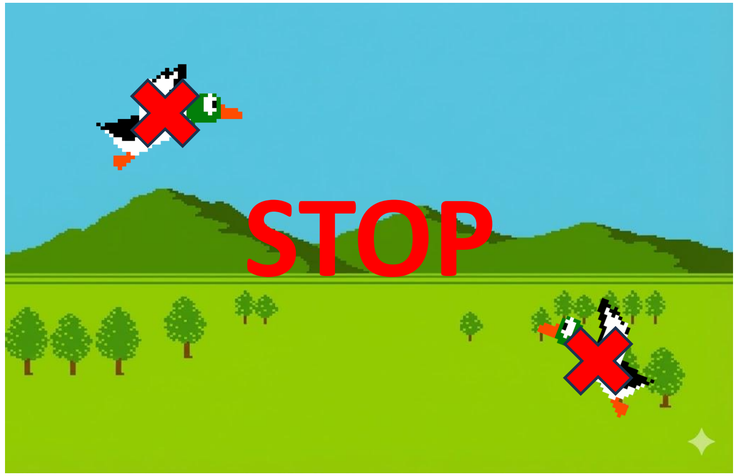
```
end else if (duck1_active && !duck1_exploding) begin
    score <= score + 1;
    hit1_pulse_cnt <= 8'd100;
end else if (duck2_active && !duck2_exploding) begin
    score <= score + 1;
    hit2_pulse_cnt <= 8'd100;
end
```

hit_pulse_cnt의 값을 15 → 100으로 증가시켜서 hit 신호 충분히 감지할 수 있도록 수정함

part 05

Trouble Shooting

일시정지 버튼 입력시 오리 소멸 문제



- 문제 : STOP 버튼을 누르면 게임이 정지는 되지만 모니터 내에 존재하던 오리들이 사라짐
- 원인 : 오리 이동 로직에 pause 체크가 없어서 화면은 멈춰있지만 오리는 계속 이동하는 중임
- 해결방안 :

```
S_FLY: begin
if (hit) begin
    state <= S_BOOM;
    boom_timer <= 0;
end else begin
    if (speed_cnt < current_speed_limit)
        speed_cnt <= speed_cnt + 1;
    else begin
        speed_cnt <= 0;
        case (spawn_type)
            0: begin
                if (!is_facing_left) begin
                    if (duck_x < H_RES - DUCK_W)
                        duck_x <= duck_x + 1;
                    else is_facing_left <= 1;
                end else begin
                    if (duck_x > 0) duck_x <= duck_x - 1;
                    else if (!pause) state <= S_IDLE;
                end
            end
```

```
S_FLY: begin
    if (hit) begin
        state <= S_BOOM;
        boom_timer <= 0;
    end else if (!pause) begin  // pause 체크 추가
        if (speed_cnt < current_speed_limit)
            speed_cnt <= speed_cnt + 1;
        else begin
            speed_cnt <= 0;
            case (spawn_type)
                0: begin
                    if (!is_facing_left) begin
                        if (duck_x < H_RES - DUCK_W)
                            duck_x <= duck_x + 1;
                        else is_facing_left <= 1;
                    end else begin
                        if (duck_x > 0) duck_x <= duck_x - 1;
                        else state <= S_IDLE;
                    end
                end
```

part 06

역할 및 고찰

김문호

역할 : QQQVGA 데이터 출력을 위한 OV7670 설정 ROM 제작
영상 데이터 입출력 시스템 제작

고찰 : 게임 구현을 위한 이미지 ROM을 많이 사용하면서 BRAM 용량이 부족해졌고 영상 데이터 ROM의 최적화가 필요했습니다.
이 문제를 해결하면서 사용하는 Chip의 Resource를 고려한 설계의 중요성을 느꼈습니다.

김태형

역할 : 오리 픽셀 데이터 작업 / 게임 구조 확립
오리의 이동, 애니메이션 로직 설계, 격추 동작 생성

고찰 : VGA를 이용한 영상처리 로직을 설계하며 데이터 메커니즘에 대한 이해를 얻었고 간단한 픽셀 게임이지만 실시간으로 영상을 처리하는 게임을 완성했다는 것에 대한 큰 성취감을 느꼈습니다.

이재원

역할 : 오리 포착 시스템 구현, 각종 버튼 기능 구현, 발표 담당

고찰 : Color 감지 기능을 설계하면서, 코드의 색상값과 카메라가 실제로 감지하는 색상에는 차이가 있다는 것을 알게 되었고, 직접 수치를 바꿔가며 테스트하여 적절한 값을 찾을 수 있었습니다. 하드웨어 개발에서는 실제 환경에 맞춰 설계하는 과정이 중요하다는 것을 깨닫게 되었습니다.

채준희

역할 : 카메라화면/게임화면 결합
권총게임기 제작,오리 픽셀 데이터 작업

고찰 : 고전게임, 영상처리, 디스플레이, 버튼에 사용한 회로 등 머릿속 다른 카테고리에 있던 지식들을 한 프로젝트에 녹여내는 과정을 통해 즐거움과 성취감을 느꼈습니다.

Thank You