Mini Golf 🚩

Object Drawing

```
-- Drawing

draw_proc : PROCESS(pixel_row, pixel_col, ball_x, ball_y,
                    shot_state, aim_vx, aim_vy)
    VARIABLE px, py   : INTEGER;
    VARIABLE dx, dy   : INTEGER;
    VARIABLE r2       : INTEGER;
    VARIABLE arrow_x  : INTEGER;
    VARIABLE arrow_y  : INTEGER;
    VARIABLE adx, ady : INTEGER;
    VARIABLE ar2      : INTEGER;
BEGIN
    px := CONV_INTEGER(pixel_col);
    py := CONV_INTEGER(pixel_row);

    wall_on <= '0';
    water_on <= '0';
    ball_on <= '0';
    hole_on <= '0';
    arrow_on <= '0';

    -- box walls
    IF ((px >= BOX_LEFT) AND (px <= BOX_RIGHT) AND
        (py >= BOX_TOP) AND (py <= BOX_TOP + 5)) OR
       ((px >= BOX_LEFT) AND (px <= BOX_RIGHT) AND
        (py >= BOX_BOTTOM - 5) AND (py <= BOX_BOTTOM)) OR
       ((py >= BOX_TOP) AND (py <= BOX_BOTTOM) AND
        (px >= BOX_LEFT) AND (px <= BOX_LEFT + 5)) OR
       ((py >= BOX_TOP) AND (py <= BOX_BOTTOM) AND
        (px >= BOX_RIGHT - 5) AND (px <= BOX_RIGHT)) THEN
        wall_on <= '1';
    END IF;

    -- water
    IF (px >= WATER_LEFT) AND (px <= WATER_RIGHT) AND
       (py >= WATER_TOP)  AND (py <= WATER_BOTTOM) THEN
        water_on <= '1';
    END IF;

    -- ball
    dx := ball_x - px; IF dx < 0 THEN dx := -dx; END IF;
    dy := ball_y - py; IF dy < 0 THEN dy := -dy; END IF;
    r2 := dx*dx + dy*dy;
    IF r2 <= BALL_R*BALL_R THEN
        ball_on <= '1';
```
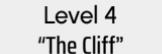
Level Instances

```
L1_core : ENTITY work.minigolf_level1
PORT MAP(
    v_sync => s_vsync, pixel_row => s_pixel_row, pixel_col => s_pixel_col,
    BTNU => BTNU, BTND => BTND, BTNL => BTNL, BTNR => BTNR, BTNC => BTNC,
    level_active => act1,
    red => L1_r, green => L1_g, blue => L1_b,
    stroke_pulse => L1_stroke, hole_pulse => L1_hole
);

L2_core : ENTITY work.minigolf_level2
PORT MAP(
    v_sync => s_vsync, pixel_row => s_pixel_row, pixel_col => s_pixel_col,
    BTNU => BTNU, BTND => BTND, BTNL => BTNL, BTNR => BTNR, BTNC => BTNC,
    level_active => act2,
    red => L2_r, green => L2_g, blue => L2_b,
    stroke_pulse => L2_stroke, hole_pulse => L2_hole
);

L3_core : ENTITY work.minigolf_level3
PORT MAP(
    v_sync => s_vsync, pixel_row => s_pixel_row, pixel_col => s_pixel_col,
```

Score Tracking

```
-------------------------------------------------------
-- Score tracking (TOTAL + per-hole) + win/restart
-------------------------------------------------------

score_proc : PROCESS(clk_in)
    VARIABLE idx : INTEGER;
BEGIN
    IF rising_edge(clk_in) THEN
        IF (game_won='1') THEN
            IF btnc_rise='1' THEN
                game_won <= '0';
                level_state <= to_unsigned(0,3);
                total_strokes <= (others => '0');
                cur_hole_strokes <= (others => '0');
                hole_strokes <= (others => (others => '0'));
            END IF;

        ELSE
            IF stroke_rise='1' THEN
                total_strokes <= total_strokes + 1;
                cur_hole_strokes <= cur_hole_strokes + 1;
            END IF;

            IF hole_rise='1' THEN
                idx := to_integer(level_state);
                IF idx >= 0 AND idx <= 4 THEN
                    hole_strokes(idx) <= cur_hole_strokes;
                END IF;
                cur_hole_strokes <= (others => '0');

                IF level_state = to_unsigned(4,3) THEN
                    game_won <= '1';
                ELSE
                    level_state <= level_state + 1;
                END IF;
            END IF;
```
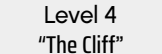
Level Progression

```
-- Level active decoding (static signals for port maps)

act1 <= '1' WHEN (game_won='0' AND level_state = to_unsigned(0,3) ELSE '0';
act2 <= '1' WHEN (game_won='0' AND level_state = to_unsigned(1,3) ELSE '0';
act3 <= '1' WHEN (game_won='0' AND level_state = to_unsigned(2,3) ELSE '0';
act4 <= '1' WHEN (game_won='0' AND level_state = to_unsigned(3,3) ELSE '0';
act5 <= '1' WHEN (game_won='0' AND level_state = to_unsigned(4,3) ELSE '0';
```

minigolf.vhd | **minigolf_level1.vhd** | minigolf_level2.vhd | minigolf_level3.vhd | minigolf_level4.vhd | minigolf_level5.vhd

```
-- hole detection
dx := nx - HOLE_X; IF dx < 0 THEN dx := -dx; END IF;
dy := ny - HOLE_Y; IF dy < 0 THEN dy := -dy; END IF;
r2 := dx*dx + dy*dy;

IF r2 <= HOLE_R*HOLE_R THEN
    hole_pulse_i <= '1';
    shot_state   <= '0';
    ball_x       <= START_X;
    ball_y       <= START_Y;
ELSE
    ball_x <= nx;
    ball_y <= ny;
    shot_timer <= shot_timer + 1;
    IF shot_timer >= MAX_STEPS THEN
        shot_state <= '0';
```

Level 1
"First Steps"

Level 2
"The Maze"

Level 3
"Magnets"

Level 4
"The Cliff"

Level 5
"Take Flight"

With every stroke you make, the counter increases by 1. Try to get the lowest score that you can!

Use the board's built in buttons to control the ball's shot power, direction, and to shoot the ball.

Ball Movement

```
-- BALL MOVING
nx := ball_x + vel_x;
ny := ball_y + vel_y;

-- bounce off box walls
IF nx <= BOX_LEFT + BALL_R THEN
    nx   := BOX_LEFT + BALL_R;
    vel_x <= -vel_x;
ELSIF nx >= BOX_RIGHT - BALL_R THEN
    nx   := BOX_RIGHT - BALL_R;
    vel_x <= -vel_x;
END IF;

IF ny <= BOX_TOP + BALL_R THEN
    ny   := BOX_TOP + BALL_R;
    vel_y <= -vel_y;
ELSIF ny >= BOX_BOTTOM - BALL_R THEN
    ny   := BOX_BOTTOM - BALL_R;
    vel_y <= -vel_y;
END IF;
```

Level Generation

```
ARCHITECTURE Behavioral OF minigolf_level1
-- Course layout constants
CONSTANT H_RES      : INTEGER := 800;
CONSTANT V_RES      : INTEGER := 600;

-- water in the middle
CONSTANT WATER_LEFT    : INTEGER := 360;    CONSTANT BOX_LEFT   : INTEGER := 100;
CONSTANT WATER_RIGHT   : INTEGER := 440;    CONSTANT BOX_RIGHT  : INTEGER := 700;
CONSTANT WATER_TOP     : INTEGER := 260;    CONSTANT BOX_TOP    : INTEGER := 100;
CONSTANT WATER_BOTTOM  : INTEGER := 340;    CONSTANT BOX_BOTTOM : INTEGER := 500;

-- aiming "dot"                             CONSTANT BALL_R     : INTEGER := 6;
CONSTANT ARROW_SCALE : INTEGER := 4;        CONSTANT HOLE_R     : INTEGER := 10;
CONSTANT ARROW_R     : INTEGER := 4;
                                            CONSTANT START_X    : INTEGER := 150;
                                            CONSTANT START_Y    : INTEGER := 300;
CONSTANT MAX_STEPS   : INTEGER := 80;
                                            CONSTANT HOLE_X     : INTEGER := 650;
                                            CONSTANT HOLE_Y     : INTEGER := 300;
-- valley attraction zone
                                            CONSTANT VALLEY_R   : INTEGER := 80;
-- Positions & state

SIGNAL ball_x       : INTEGER RANGE 0 TO H_RES-1 := START_X;
SIGNAL ball_y       : INTEGER RANGE 0 TO V_RES-1 := START_Y;

SIGNAL aim_vx       : INTEGER RANGE -16 TO 16 := 4;
SIGNAL aim_vy       : INTEGER RANGE -16 TO 16 := 0;

SIGNAL vel_x        : INTEGER RANGE -32 TO 32 := 0;
SIGNAL vel_y        : INTEGER RANGE -32 TO 32 := 0;
```