```cpp
#include <imtui/imtui.h>
#include <imtui/imtui-impl-ncurses.h>
#include <ctime>

using namespace ImGui;

float ball_x = 0.0f, ball_y = 0.0f;
float ball_vel_x = 1.0f, ball_vel_y = 1.0f;

constexpr float WIDTH = 60.0f;
constexpr float HEIGHT = 30.0f;
constexpr float RADIUS = 1.2;

void init_ball() {
    ball_x = rand() % (int)WIDTH;
    ball_y = rand() % (int)HEIGHT;
    ball_vel_x = rand() % 3 + 1;
    ball_vel_y = rand() % 3 + 1;
}

// Function to update ball position and handle bouncing
void update_ball() {
    // Update position
    ball_x += ball_vel_x;
    ball_y += ball_vel_y;

    // Bounce off horizontal walls
    if (ball_x  <= 0.0f || ball_x + RADIUS >= WIDTH) {
        ball_vel_x = -ball_vel_x;
        // ball_x = (ball_x - RADIUS <= 0.0f) ? RADIUS : WIDTH - RADIUS;
    }

    // Bounce off vertical walls
    if (ball_y <= 0.0f || ball_y + RADIUS >= HEIGHT) {
        ball_vel_y = -ball_vel_y;
        // ball_y = (ball_y - RADIUS <= 0.0f) ? RADIUS : HEIGHT - RADIUS
;
    }
}

void draw_ball(ImDrawList* draw_list, ImVec2 canvas_pos) {
    ImVec2 ball_center = ImVec2(canvas_pos.x + ball_x,
                                canvas_pos.y + ball_y);
    draw_list->AddRectFilled(ball_center,
        ImVec2(ball_center.x + RADIUS, ball_center.y + RADIUS),
        IM_COL32(0, 255, 0, 255));
}

// Function to render the game area
// Returns true if we should exit
bool render_game() {
    SetNextWindowPos(ImVec2(2, 2), ImGuiCond_Once);
    SetNextWindowSize(ImVec2(WIDTH , HEIGHT + 10), ImGuiCond_Once);

    Begin("Bouncing Ball (Procedural - Circle)");

    // Get the draw list for custom drawing
    ImDrawList* draw_list = GetWindowDrawList();
    ImVec2 canvas_pos = GetCursorScreenPos();
    ImVec2 canvas_size = ImVec2(WIDTH, HEIGHT);

    // // Draw border rectangle
    // draw_list->AddRect(canvas_pos,
    //     ImVec2(canvas_pos.x + canvas_size.x,
    //            canvas_pos.y + canvas_size.y),
    //     IM_COL32(255, 255, 255, 255), 0.0f, 0, 1.0f);

    draw_ball(draw_list, canvas_pos);


    // Reserve space for the canvas
    Dummy(canvas_size);

    // Display info
    Text("Position: (%.1f, %.1f)", ball_x, ball_y);
    Text("Velocity: (%.1f, %.1f)", ball_vel_x, ball_vel_y);
    Text("Radius: %.1f", RADIUS);
    Text("Canvas Pos: %.1f %.1f", canvas_pos.x, canvas_pos.y);


    bool should_exit = false;
    if (Button("Exit")) {
        should_exit = true;
    }

    End();

    return should_exit;
}

int main() {
    srand(time(NULL));
    IMGUI_CHECKVERSION();
    CreateContext();

    auto screen = ImTui_ImplNcurses_Init(true);
    ImTui_ImplText_Init();

    // Initialize ball
    init_ball();

    bool should_exit = false;
    while (!should_exit) {
        ImTui_ImplNcurses_NewFrame();
        ImTui_ImplText_NewFrame();

        NewFrame();

        // Update ball
        update_ball();

        // Render game
        should_exit = render_game();

        Render();

        ImTui_ImplText_RenderDrawData(GetDrawData(), screen);
        ImTui_ImplNcurses_DrawScreen();
    }

    ImTui_ImplText_Shutdown();
    ImTui_ImplNcurses_Shutdown();

    return 0;
```

```
}
```