Programação com a linguagem Céu

Eventos internos

Anny Caroline
annycarolinegnr@gmail.com

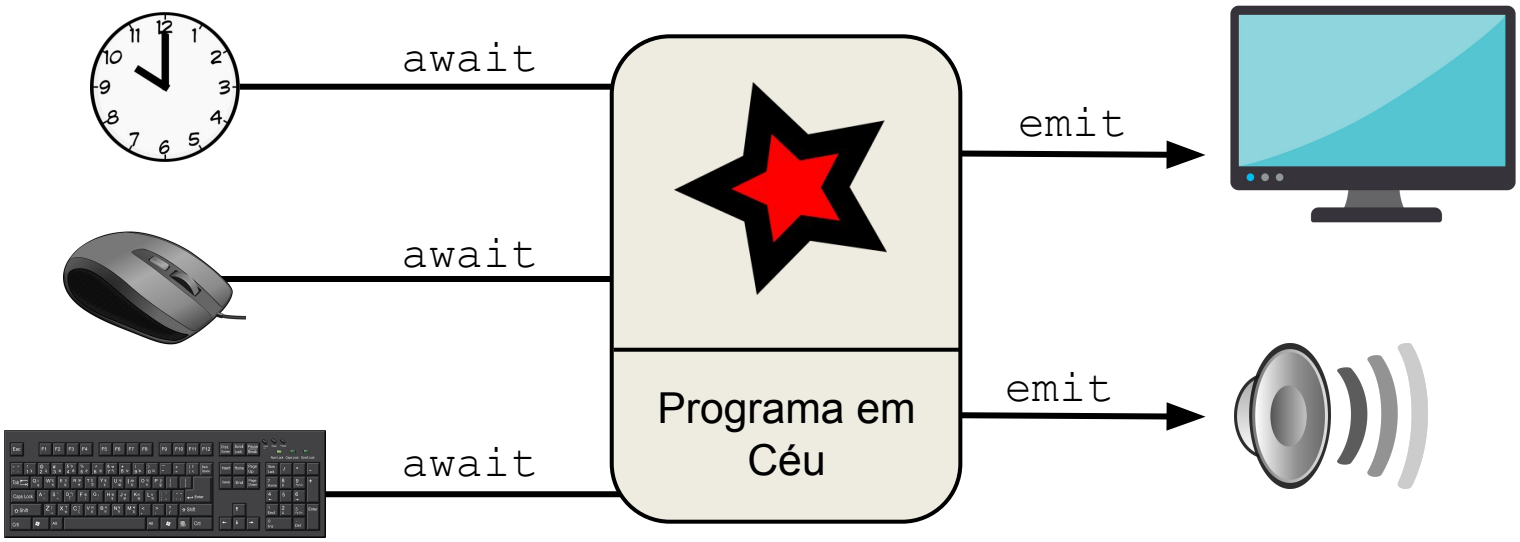
Francisco Sant'Anna
francisco@ime.uerj.br

Continuando nosso exemplo

- Discutimos no slide anterior o `code/await Player` deve ser responsável por atualizar e desenhar sua pontuação
- Já conseguimos desenhar, mas a atualização ainda está no "programa principal"
- O ideal seria que o programa principal pudesse indicar quando uma colisão ocorresse, e que todos os interessados reagissem a essa colisão
  - para isso usaremos um **evento interno**

Mas o que é um evento interno?

- Até agora usamos somente eventos externos
- Os eventos externos permitem que nosso programa se comunique com o mundo externo, por exemplo, emitindo um evento de desenho ou de áudio, ou aguardando um evento de clique de tecla ou passagem de tempo



Mas o que é um evento interno?

- Já os eventos internos são usados na comunicação interna do nosso programa, permitindo que parte do código se comuniquem com outras

```
event none draw;

par do
    await draw;
    emit GRAPHICS_DRAW_PIXEL(0,0);
with
    await MOUSE_CLICK;
    emit draw;
end
```

Continuando nosso exemplo

- Vamos começar declarando um evento interno logo no começo do programa

```
#include "random.ceu"
emit FRAMES_SET(yes);

event none collision;

//...
```

1-event.ceu

```
// Aqui vamos tratar a colisão
every FRAMES_UPDATE do
    var& Player player;
    loop player in players do
        var&? Pixel pixel;
        loop pixel in pixels do
            // verifico a colisão e tomo uma ação
            if (player.pt.x == pixel!.pt.x) and
                (player.pt.y == pixel!.pt.y) then
                kill pixel;
                emit collision;
            end
        end
    end
end
```

1-event.ceu

```
code/await Player(var int id, var int up, var int right, var int
down, var int left, var Color color) -> (var Point pt) -> NEVER do
    var int score = 0;    pt = val Point(0,0);
    var int x_axis = 0;   var int y_axis = 0;

    par do  //every key in KEY_PRESS do
    with    // every key in KEY_UNPRESS do
    with    // every FRAMES_UPDATE do
    with    // every FRAMES_REDRAW do
    with
        loop do
            await outer.collision;
            score = score + 1;
            _printf("%d - score %d\n", id, score);
        end
    end
end
```

1-event.ceu

Ainda não está correto

- Como os dois `Players` aguardam o evento `collision`
- Cada vez que ele é emitido, isto é, a cada colisão, os dois `Players` são notificados
- Console, após 3 colisões

```
1 - score 1
2 - score 1
1 - score 2
2 - score 2
1 - score 3
2 - score 3
```

Como resolver?

- Eventos podem carregar valores

```
event int evento1;
event bool evento2;
event (int, int) evento3;
```

- Vamos fazer o evento `collision` carregar o id do `Player`

```
event int collision;
```

2-event.ceu

Como resolver?

- Enviar o id pelo evento quando houver uma colisão

```
// Aqui vamos tratar a colisão
every FRAMES_UPDATE do
    var& Player player;
    loop player in players do
        var&? Pixel pixel;
        loop pixel in pixels do
            // verifico a colisão e tomo uma ação
            if (player.pt.x == pixel!.pt.x) and (player.pt.y == pixel!.pt.y) then
                kill pixel;
                emit collision(player.id);
            end
        end
    end
end
```

Como resolver?

- Por último, vamos aguardar o evento no `code/await Player` comparando o id recebido com o já presente no `Player`.

```
loop do
    var int r_id = await outer.collision until r_id == id;
    score = score+1;
    _printf("%d - score %d\n", id, score);
end
```

2-event.ceu

Exercício

- Implemente o término do jogo

- O jogo termina quando um jogador completa 3 pontos
- Ao terminar, o jogo deve limpar a tela e exibir o jogador vencedor

- Dica:
  - Utilize um evento interno para indicar quando o jogo termina

Solução

```
#include "random.ceu"

emit FRAMES_SET(yes);

event int collision;
event int game_over;
```

3-e-event.ceu

```
var int winner;

par/or do
    every 1s do
        spawn Pixel() in pixels;
    end
with
    // Aqui vamos tratar a colisão
    every FRAMES_UPDATE do
        var&? Player player;
        loop player in players do
            var&? Pixel pixel;
            loop pixel in pixels do
                // verifico a colisão e tomo uma ação
                if (player!.pt.x == pixel!.pt.x) and (player!.pt.y == pixel!.pt.y) then
                    kill pixel;
                    emit collision(player!.id);
                end
            end
        end
    end
with
    winner = await game_over;
end

emit GRAPHICS_SET_COLOR_NAME(COLOR_WHITE);
every FRAMES_REDRAW do
    emit GRAPHICS_DRAW_INT(0,10,winner);
    emit GRAPHICS_DRAW_TEXT(0,0,"Fim");
end
```

Solução

```
code/await Pixel (none) -> (var Point pt) -> none do
    pt = call Random_Point_In_Square(-25,-25,25,17);

    var Color color = COLOR_RED;
    par/or do //loop de mudança de cor
    with // every FRAMES_REDRAW do
    with // await MOUSE_CLICK;
    with
        await outer.game_over;
    end
end
```

3-e-event.ceu

```
code/await Player(var int id, var int up, var int right, var int down, var int left, var
Color color) -> (var Point pt) -> none do
    var int score = 0;
    pt = val Point(0,0);
    var int x_axis = 0; var int y_axis = 0;

    par/or do // every key in KEY_PRESS do
    with // every key in KEY_UNPRESS do
    with // every FRAMES_UPDATE do
    with // every FRAMES_REDRAW do
    with
        loop do
            var int r_id = await outer.collision until r_id == id;
            score = score+1;
            _printf("%d - score %d\n", id, score);
            if (score==3) then
                emit outer.game_over(id);
            end
        end
    with
        await outer.game_over;
    end
end
```

Outra solução

- Não vamos mais alterar os `code/await Pixel` e `Player`
- Vamos usar um bloco `do end` para definir um bloco explicitamente

```
var int winner;

do
    pool[5] Pixel pixels;
    pool[2] Player players;
    spawn Player(1, KEY_UP, KEY_RIGHT, KEY_DOWN, KEY_LEFT, COLOR_BLUE) in players;
    spawn Player(2, KEY_w , KEY_d    , KEY_s   , KEY_a   , COLOR_RED) in players;

    par/or do //every 1s do
    with // tratar a colisão
    with
        winner = await game_over;
    end

end

emit GRAPHICS_SET_COLOR_NAME(COLOR_WHITE);
every FRAMES_REDRAW do
    emit GRAPHICS_DRAW_INT(0,10,winner);
    emit GRAPHICS_DRAW_TEXT(0,0,"Fim");
end
```

Outra solução

- Os pools foram declarados dentro do bloco `do end`
- Quando um evento de `game_over` é recebido, o `par/or` termina e o código continua sua execução. Quando encontra o `end` do bloco `do end`, todo o bloco termina, juntamente com os pools e seus code/awaits
- Por fim, observe que a variável `winner` foi declarada fora do bloco `do end`, para que possa ser acessada fora dele

Substituindo o kill

- Vamos eliminar o `Pixel` que colidiu utilizando um evento interno, e não mais `kill`

Substituindo o kill

```
code/await Pixel (none) -> (var Point pt, event none collision) -> none do
    pt = call Random_Point_In_Square(-25,-25,25,17);

    var Color color = COLOR_RED;
    par/or do //loop de mudança de cor
    with // every FRAMES_REDRAW do
    with // await MOUSE_CLICK;
    with // await outer.game_over;
    with
        await collision;
    end
end
```

4-noKill.ceu

Substituindo o kill

```
// Aqui vamos tratar a colisão
every FRAMES_UPDATE do
    var&? Player player;
    loop player in players do
        var&? Pixel pixel;
        loop pixel in pixels do
            // verifico a colisão e tomo uma ação
            if (player!.pt.x == pixel!.pt.x)
            and (player!.pt.y == pixel!.pt.y) then
                emit pixel!.collision();
                emit collision(player!.id);
            end
        end
    end
end
```

4-noKill.ceu