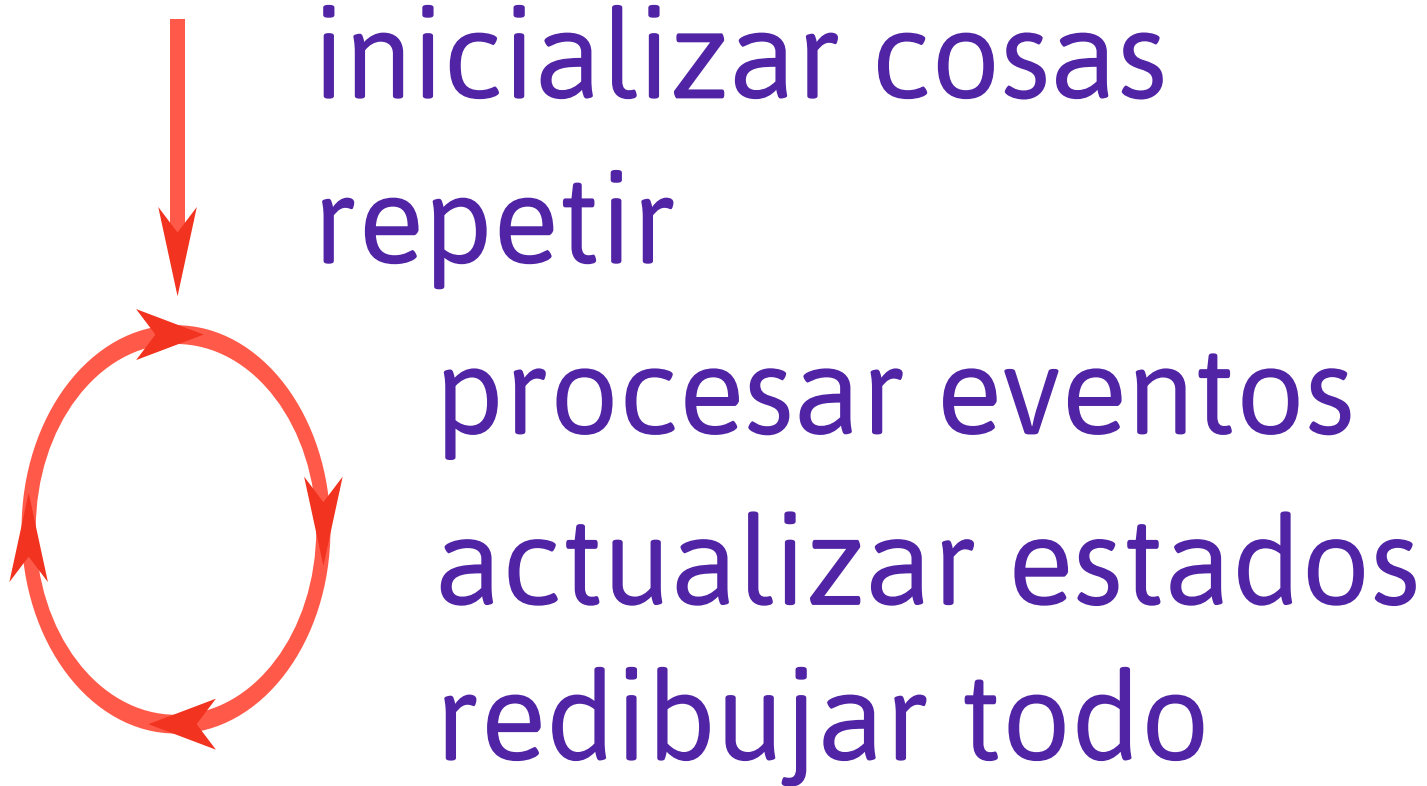


Programación Orientada a Objetos

**Unidad 9: GUIs
(parte 2: Biblioteca SFML)**

LOOP DE EVENTOS

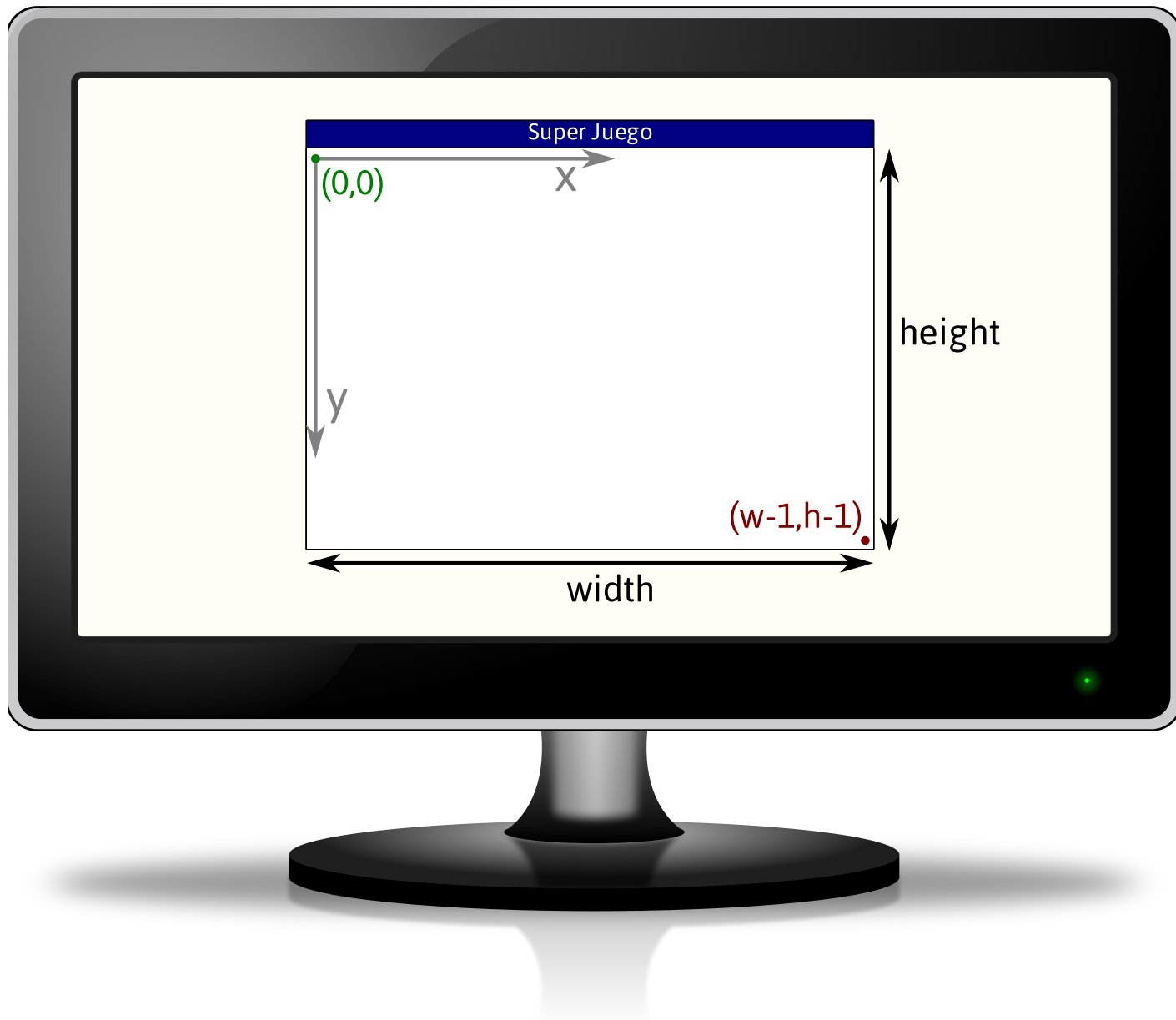


INICIALIZACIÓN

```
// crear ventana  
VideoMode vm(640, 480);  
RenderWindow win(vm, "Super Juego");
```

! a pantalla completa, no cualquier tamaño vale

SISTEMA DE COORDENADAS









✔ *Tamaños y distancias en pixeles, ángulos en grados*




INICIALIZACIÓN

```
// crear ventana  
VideoMode vm(640, 480);  
RenderWindow win(vm, "Super Juego");  
Color color_fondo(255, 255, 255, 255);
```

COLORES

Color rojo	(255, 0, 0);	
Color verde	(0, 255, 0);	
Color azul	(0, 0, 255);	

Color amarillo	(255, 255, 0);	
Color magenta	(255, 0, 255);	
Color cian	(0, 255, 255);	

Color blanco	(255, 255, 255);	
Color gris	(128, 128, 128);	
Color negro	(0, 0, 0);	

✓ El cuarto canal es para **opacidad** (0=transparente)

INICIALIZACIÓN

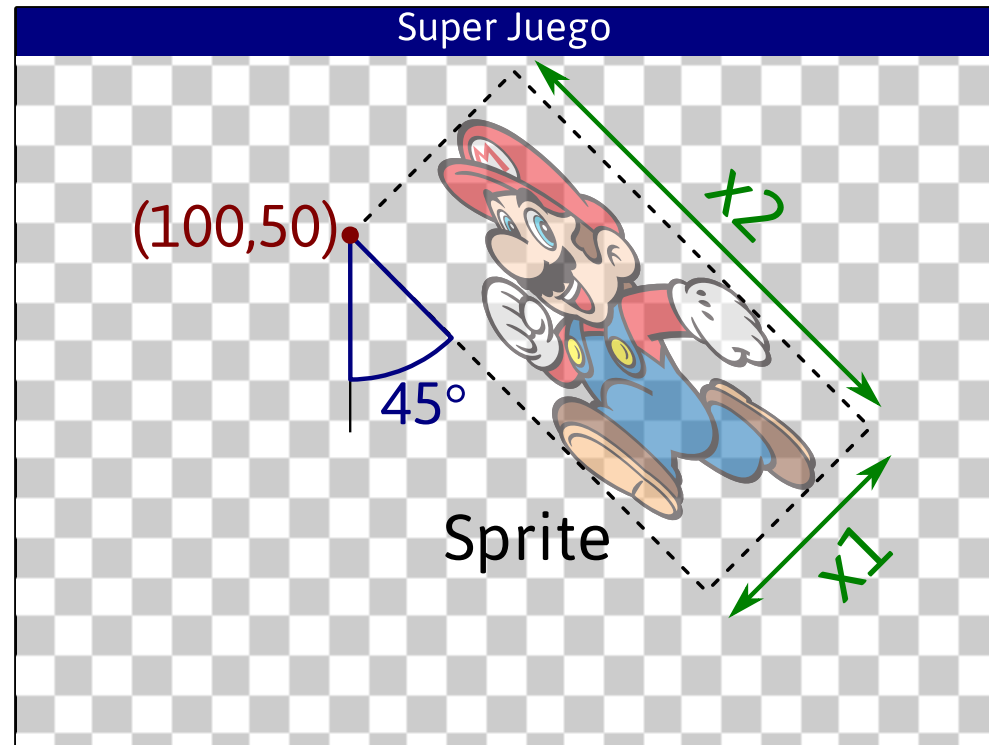
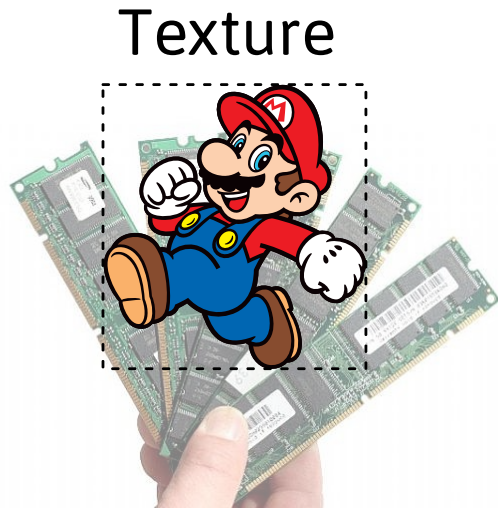
```
// crear ventana
VideoMode vm(640, 480);
RenderWindow win(vm, "Super Juego");
Color color_fondo(255, 255, 255, 255);

// cargar imagen
Texture tex;
if (! tex.loadFromFile("imagen.png") )
    cerr << "No se pudo cargar" << endl;

// definir posición inicial
Sprite spr;
spr.setTexture(tex);
spr.setPosition(100, 100);
```

! cargar una imagen (Texture) es **caro**

TEXTURE VS. SPRITE



```
spr.setPosition(100, 50);  
spr.setScale(1.0, 2.0);  
spr.setRotation(-45);  
spr.setColor(Color(255, 255, 255, 128));
```


LOOP DE EVENTOS

```
while(win.isOpen()) {  
    // procesar eventos  
    Event evt;  
    while(win.pollEvent(evt)) {  
        if(evt.type == Event::Closed)  
            win.close();  
    }  
  
    // actualizar  
    spr.move(1,1);  
  
    // redibujar todo  
    win.clear(color_fondo);  
    win.draw(spr);  
    win.display();  
}
```

❓ ¿Cuántas veces por segundo se ejecuta?

LOOP DE EVENTOS

- Velocidad fija:

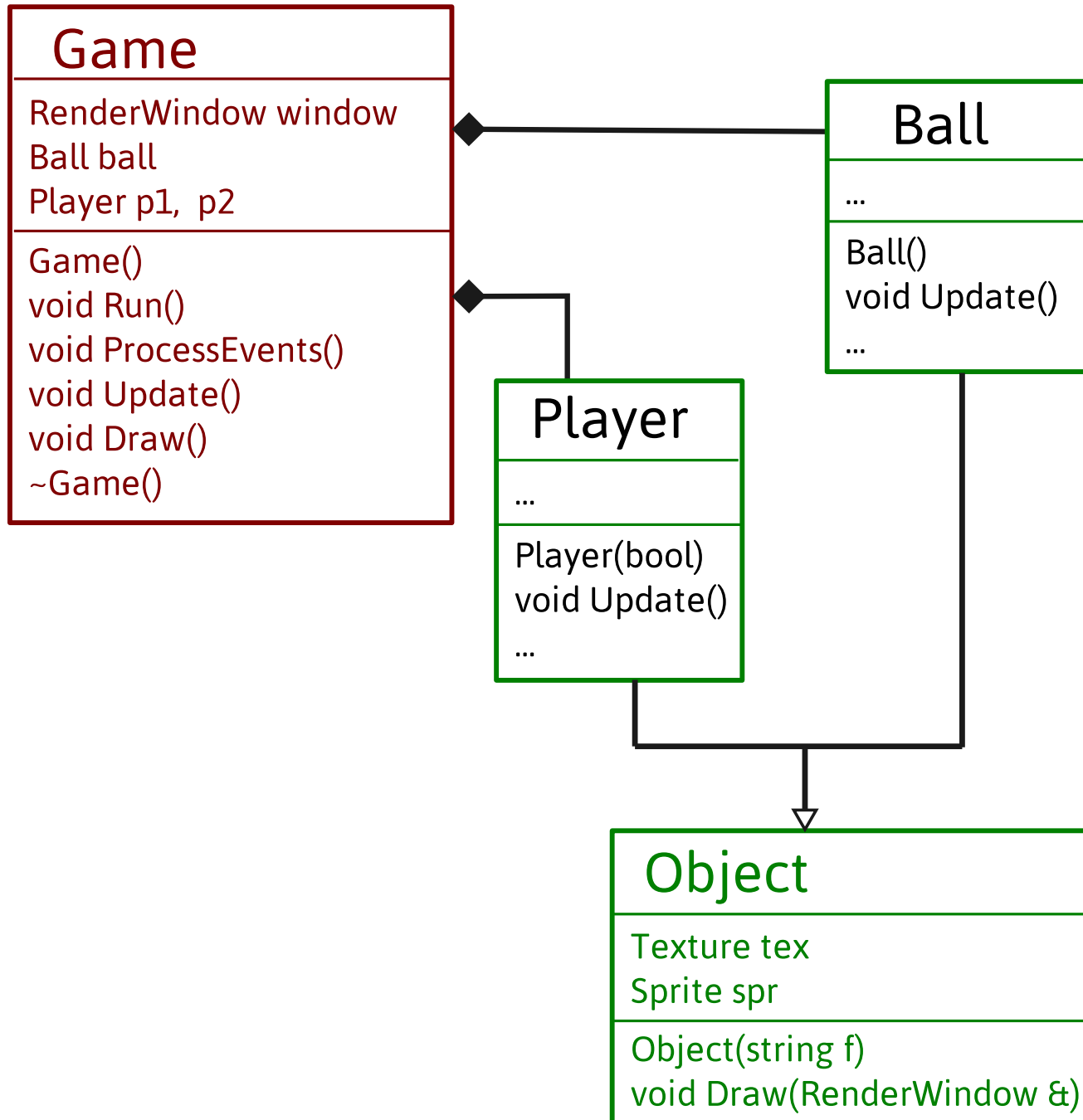
```
win.setFramerateLimit(60);  
while(...) {  
    ...actualizar según dt fijo...  
}
```

✓ 60 es un buen número para los monitores actuales

- Velocidad variable:

```
Clock clk;  
while(...) {  
    ...  
    float dt = clk.restart().asSeconds();  
    ...actualizar según dt variable...  
}
```

EJEMPLO



DETECCION DE EVENTOS

• Pooling:

```
Event evt;
while (win.pollEvent(evt)) {
    switch (evt.type) {
        case: Event::Closed:
            win.close();
            break;
        case: Event::...
            ...
    }
}
```

! no se debe omitir este loop aunque no se use

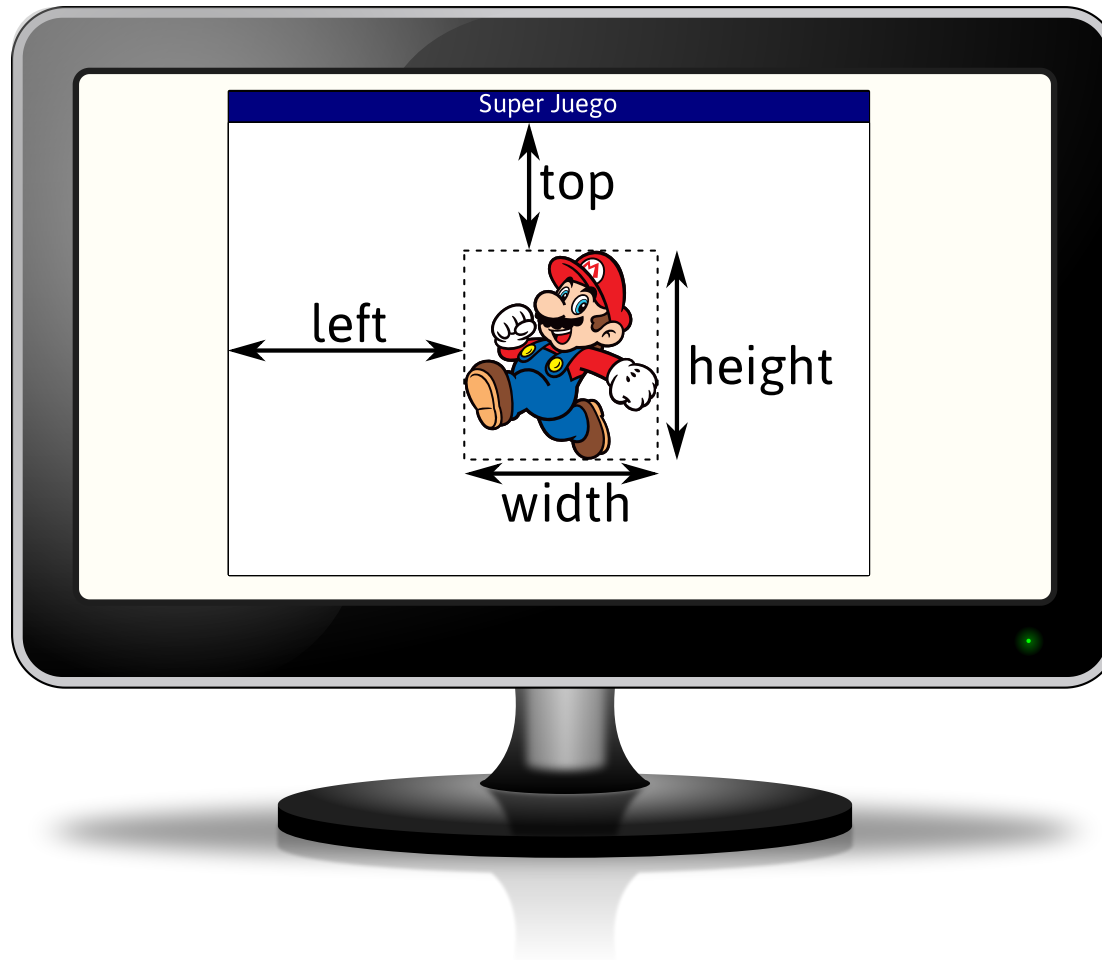
• En tiempo real:

```
if (Keyboard::isKeyPressed(Keyboard::Key::Up)) {
    ...
}
```

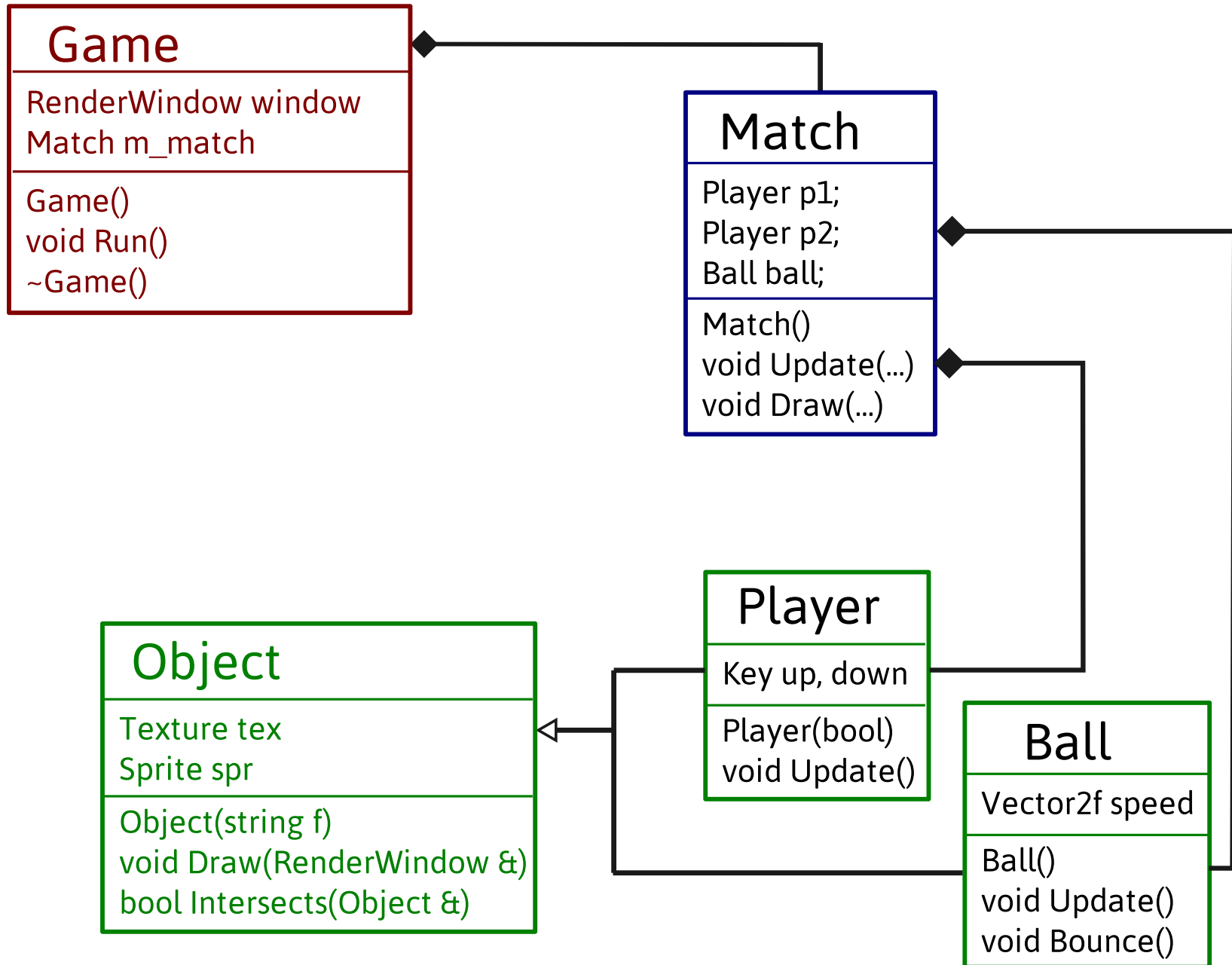
✓ método recomendado

DETECCION DE INTERSECCIONES

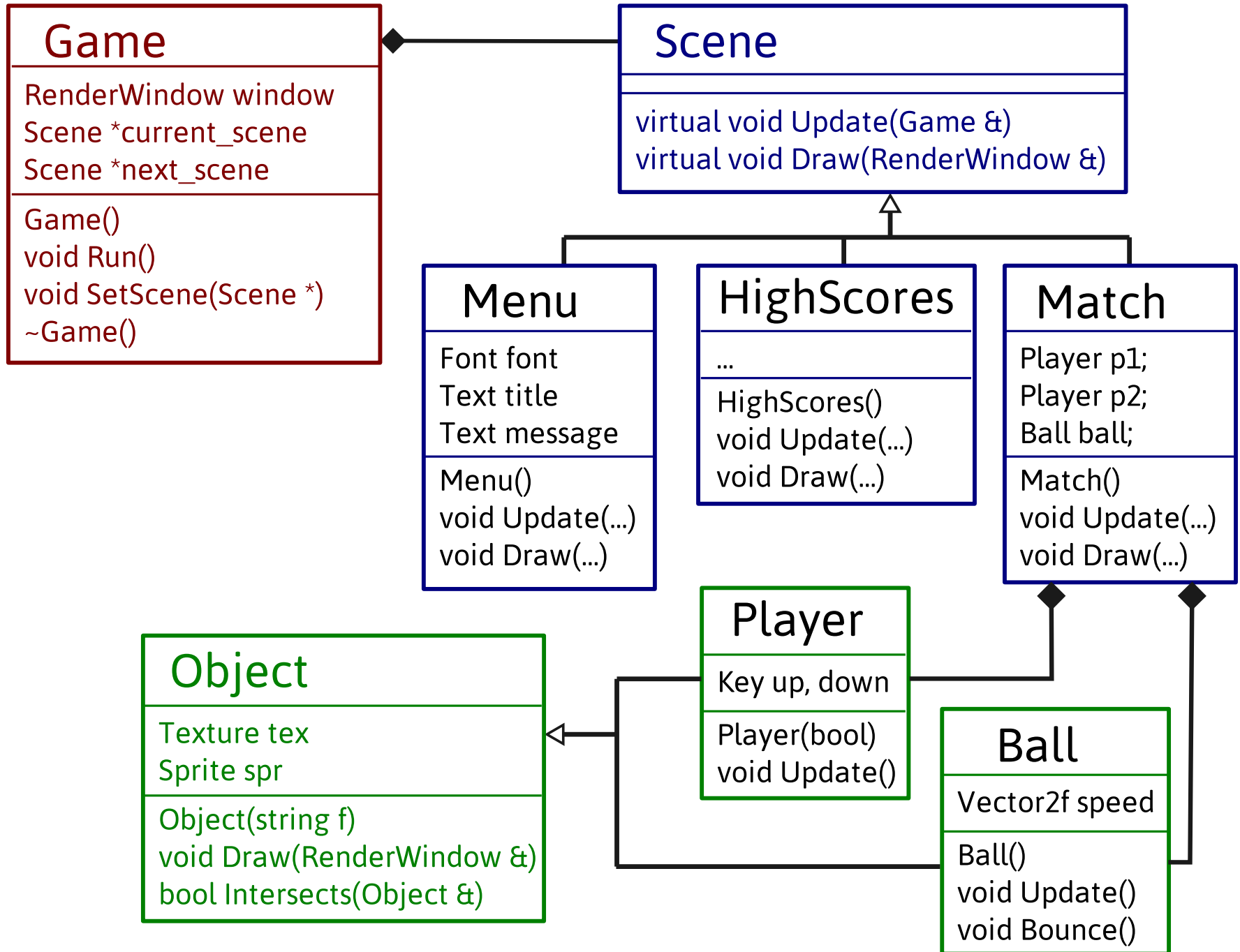
```
FloatRect r1 = sprite_1.getGlobalBounds();  
FloatRect r2 = sprite_1.getGlobalBounds();  
if (r1.intersects(r2)) { ... }
```



EJEMPLO



EJEMPLO (FINAL)



CLASE Scene

```
class Escena {  
public:  
    virtual void Update(Game &game) = 0;  
    virtual void Draw(RenderWindow &win) = 0;  
};
```

```
class Game {  
    RenderWindow m_window;  
    Escena *m_scene, *m_next_scene;  
public:  
    Game(Scene *first_scene);  
    void Run();  
    void SetScene(Scene *scene);  
    ~Game();  
};
```

```
int main() {  
    Game the_game( new EscenaMenu() );  
    the_game.Run();  
}
```


CLASE Scene

```
Game::Game(Scene *first_scene)
    : m_window(VideoMode(640, 480), "Ejemplo")
{
    m_window.setFramerateLimit(60);
    m_scene = first_scene;
    m_next_scene = nullptr;
}

void Game::SetScene (Scene * scene) {
    m_next_scene = scene;
}

Game::~~Game() {
    delete m_scene;
    delete m_next_scene;
}
```

CLASE Scene

```
void Game::Run ( ) {  
    Event evt;  
    while(m_window.isOpen()) {  
        while(m_window.pollEvent(evt)) {  
            if(evt.type == Event::Closed)  
                m_window.close();  
        }  
  
        m_scene->Update(*this);  
        m_scene->Draw(m_window);  
        m_window.display();  
  
        if (m_next_scene) {  
            delete m_scene;  
            m_scene = m_next_scene;  
            m_next_scene = nullptr;  
        }  
    }  
}
```

CLASES SFML

- ▶ Ventana: **V**ideoMode, **R**enderWindow, **V**iew
- ▶ Sprites: **T**exture, **S**prite
- ▶ Entradas: **K**eyboard, **M**ouse, **J**oystick
- ▶ Sonido: **S**oundBuffer, **S**ound, **M**usic
- ▶ Texto: **F**ont, **S**tring, **T**ext
- ▶ Formas: **C**ircleShape, **R**ectangleShape, **C**onvexShape
- ▶ Utileria: **V**ector2f, **F**loatRect, **C**olor, **C**lock
- ▶ ... y muchas más.

✓ en Zinjal, el último botón de la barra de herramientas los lleva directo a la referencia