**A**lgoritmos y programación **3** - cátedra **F**ontela

# Diseñando mi solución en POO (Galaga)


Eugenio Yolis - Mayo 2010

# Agenda

- El problema
- Solución rápida
- Limitaciones
- Buscando objetos
- Resumen
- Preguntas?

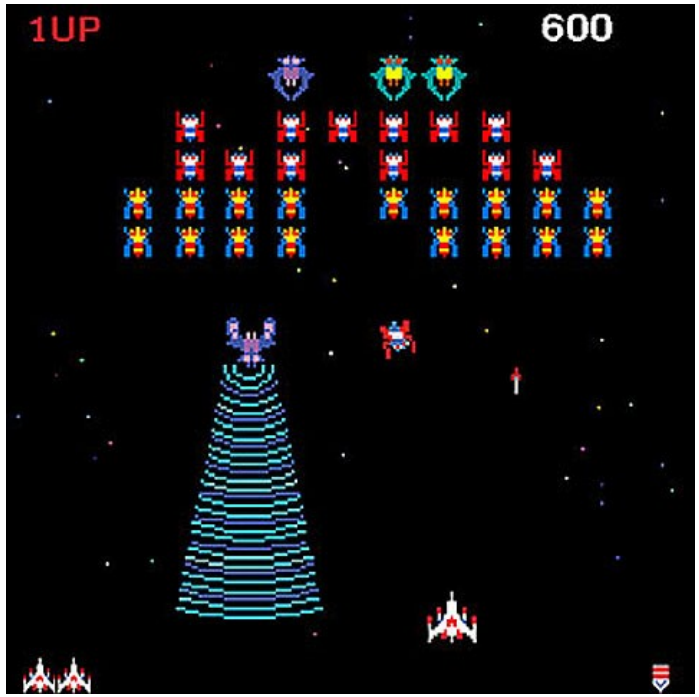
**A**lgoritmos y programación **3** - cátedra **F**ontela

# El problema

Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

# Implementar modelo del “Galaga”

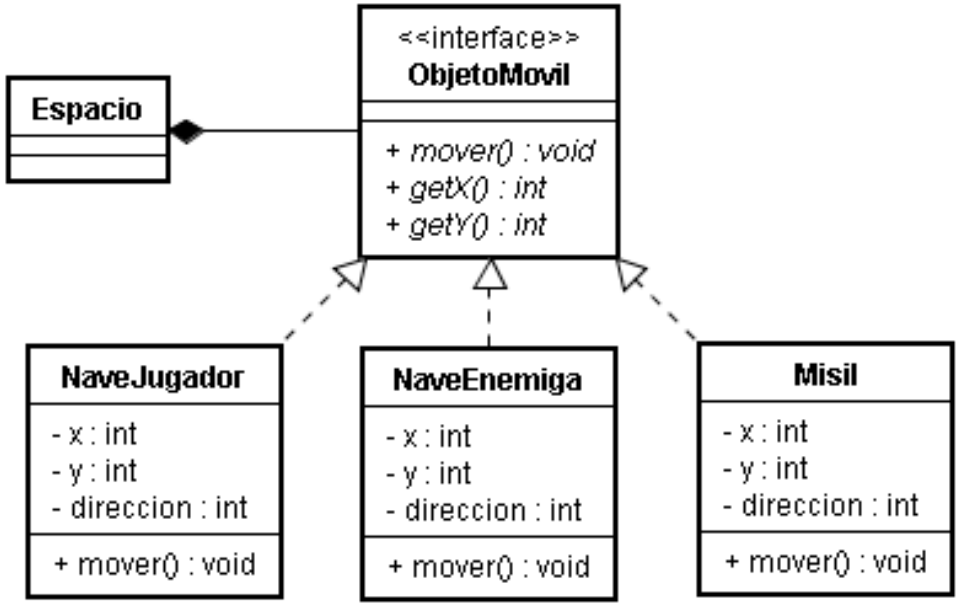
- las naves se mueven por el espacio
- el jugador maneja su nave
- la nave del usuario dispara a las otras, y las otras le disparan al él



# Solución rápida

Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

# Diagrama de clases




- creo que ya tengo el diseño base... empiezo a codificar y despues le voy agregando la funcionalidad pedida

# Ejemplos de código

```
int incX, incY;
if (this.direccion == 1) {
   incX = 0; incY = 1;  // arriba
} else if ... {  .... }
this.x = this.x + incX;
this.y = this.y + incY;
```

```
if (this.y > getEspacio().getY()) {
   // choque contra el borde superior
   this.y = this.y - 1;
}
```

```
// voy a “chocar” al jugador
int posJugY = espacio.getNaveJug().getY();
int incY = posJugY  / Math.abs(posJugY);
this.y = this.y + incY;
```

**A**lgoritmos y programación **3** - cátedra **F**ontela

# Limitaciones

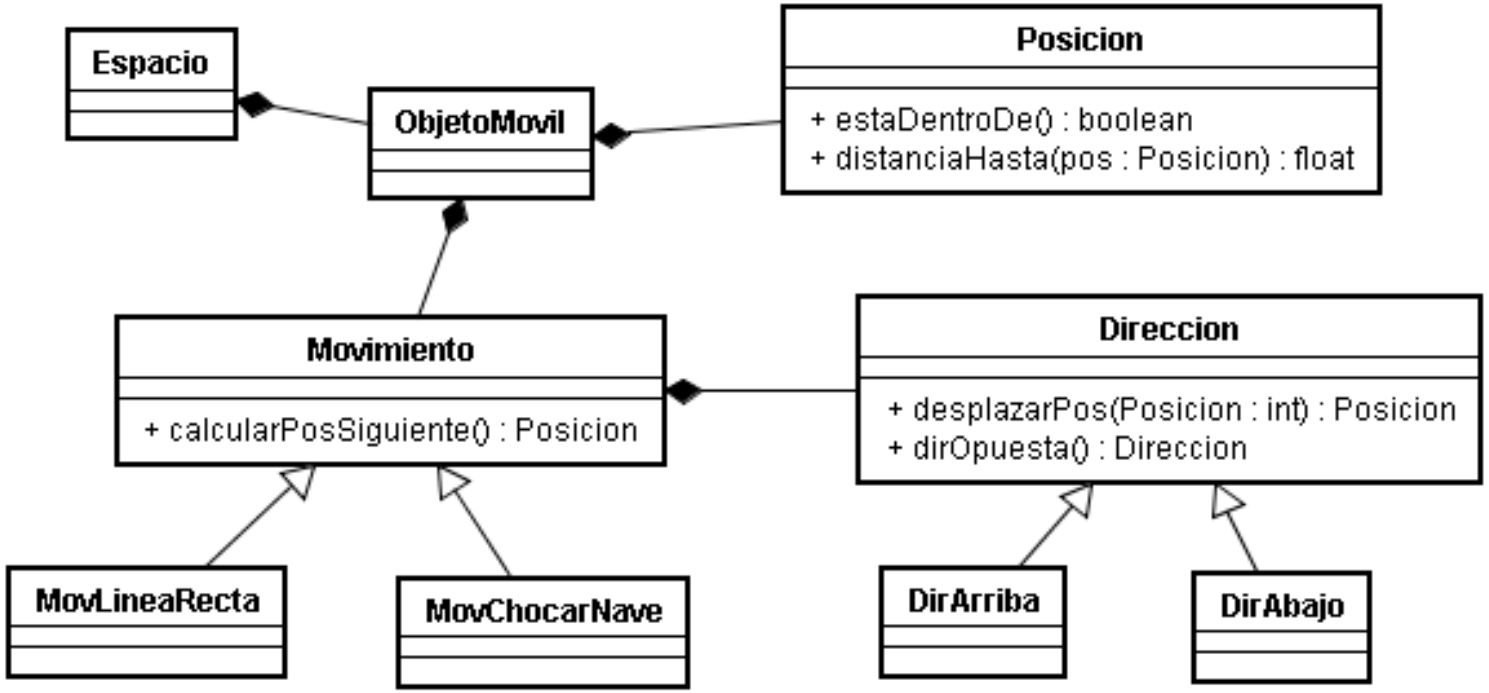
# Limitaciones

- Repetición de codigo
  - Una mejora es hacer un “ObjetoMovil” abstracto, pero igual es un problema
- Constantes magicas
  - if (this.direccion == 1)
- Calculos y condiciones dificiles de entender
  - Esto va empeorando cada vez que se agrega un nuevo caso a contemplar
- Poco encapsulamiento
  - Diferentes metodos de la clase van a manipular el “x” y el “y” directamente

**A**lgoritmos y programación **3** - cátedra **F**ontela

# Buscando objetos

# Diagrama de clases




- Las naves delegan en movimiento la logica del “mover”
- Dirección y Posición encapsulan calculos
- Diferentes naves pueden usar las mismas estrategias de movimiento

# Ejemplos de código

```
// en Nave.mover

this.posicion = this.movimiento.calcularPosSiguiente(this.posicion)
```

```
// en Movimiento.calcularPosSiguiente

posicion = this.direccion.desplazarPos(posicion)
```

```
// en Movimiento.calcularPosSiguiente

if (!posicion.dentroDe(espacio.getLimites()) {

   // choque contra algun borde del espacio

   this.direccion = direccion.dirOpuesta();

}
```

**A**lgoritmos y programación **3** - cátedra **F**ontela

# Resumen

Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

# Resumen

- Encapsular todo lo posible
- Pensar de quién es la responsabilidad
- Revisar métodos “largos”
- Revisar clases que tienen “getters” y “setters” para todos sus atributos
- Revisar el uso de condicionales y estructuras “case”
- Revisar el uso de tipos “standard” (primitivos, String, List) como parametros de entrada y de retorno

# ¿Preguntas?

Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires