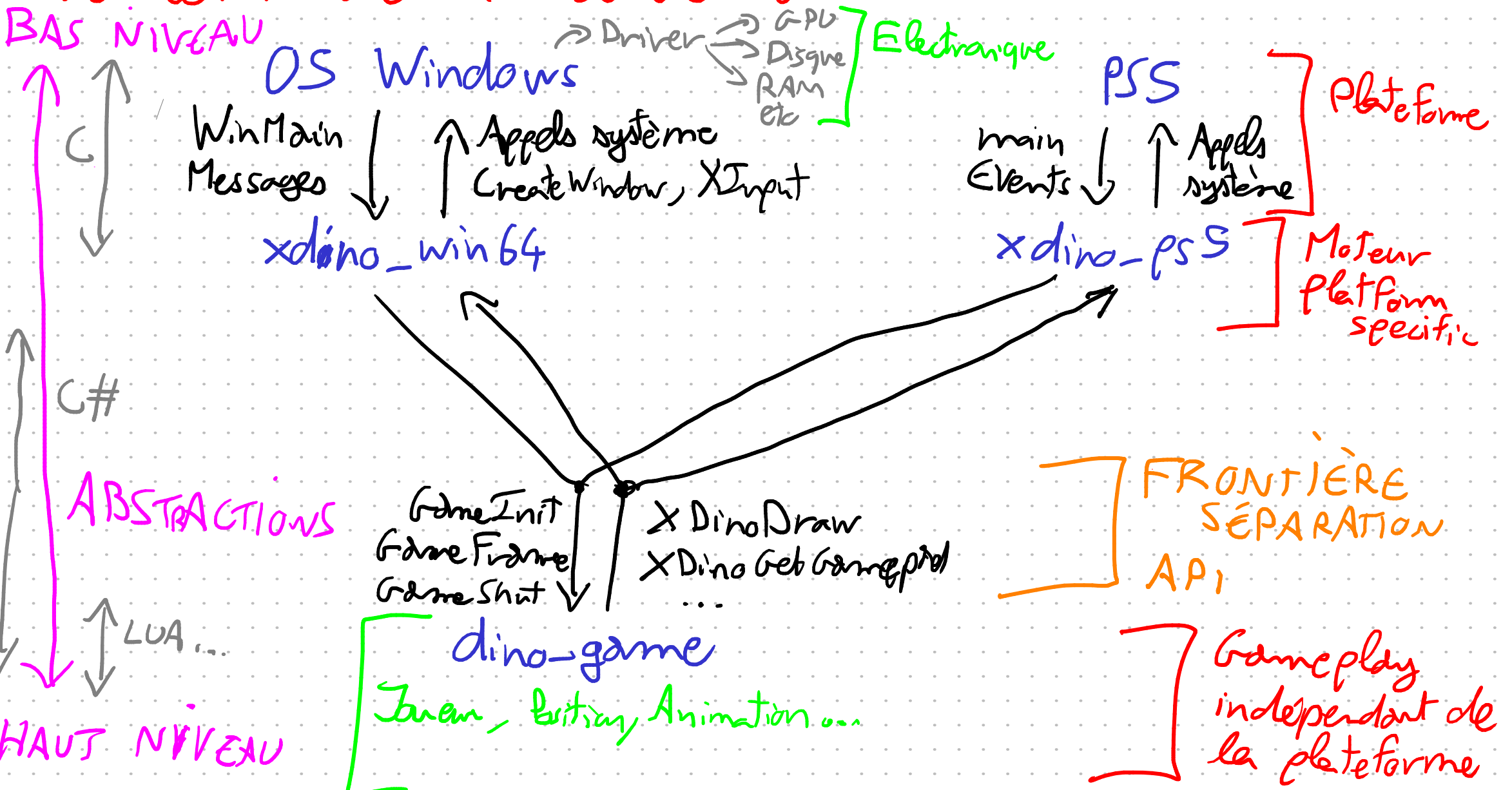


# Architecture de la base de code



AVANT

GameInit

GameFrame

GameShut

VAR GLOBALES:

g-Pos

g-bMirror

) x 4

[ x 4 variables  
[ x 4 code

APRÈS

struct DinoPlayer {

pos

bMirror

bIdle

bRunning

bWalking

void UpdatePlayer();

void DrawPlayer();

}; DinoPlayer p1, p2, p3, p4;

void DinoPlayer::UpdatePlayer (

void DinoPlayer::DrawPlayer (

player 1. UpdatePlayer (—)

this → pos

this → bMirror

ou

pos

bMirror

CE QU'ON A FAÏT

```
void DinoPlayer::UpdatePlayer()  
void DinoPlayer::DrawPlayer()  
    MEMBRE this -> pos.x += 10 IMPLICITE
```

FUNCTIONS MEMBRES  
= MÉTHODES

CE QU'ON AURAIT PU FAIRE

```
void UpdatePlayer(DinoPlayer* p, —)  
    p -> pos.x += 10 EXPLICITE
```

(Par pointeur pour ne pas  
copier)  
FUNCTIONS LIBRES

⇔ ÉQUIVALENT

# struct VS classes

(C++ s'appelait "C with classes")

Une classe est comme une structure (en C++), sauf qu'on rajoute une frontière, une séparation, entre le comportement et les détails d'implémentation.

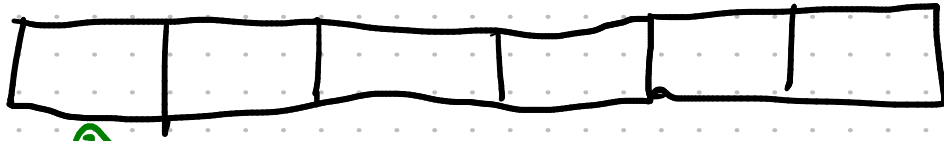
[ex: comportement = je me déplace  
détail = le sprite choisi pour l'anim]

```
class DinoPlayer {  
public:  
    void UpdatePlayer();  
    void DrawPlayer();  
    void Init (color, posInit);
```

Public = comportement  
Private = implémentation

```
private:  
    m_pos; m_bMirror  
    m_bWalking m_bRunning  
    m_bIdle  
    m_color  
};
```

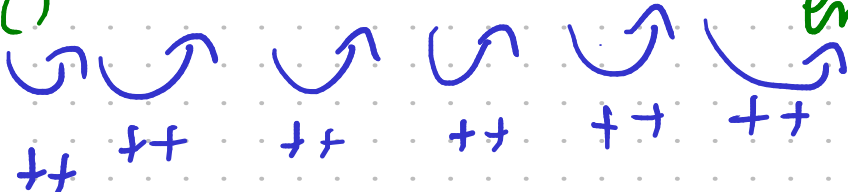
# std::vector et les algorithmes C++



begin()

end()

Itérateurs



```
VERSION FOR-RANGE
for (int& i : vec) {
    i = ...
}
```

```
VERSION INDICE
for (int i = 0; i < vec.size(); ++i) {
    vec[i] = ...
}
```

```
VERSION ITÉRATEUR
it = vec.begin()
itend = vec.end()
while (it != itend) {
    *it = ... } MODIFIE LA CASE RÉFÉRÉE
    it++ } CHANGE DE CASE
}
```

std::sort

plus petit au plus grand

- start  $\leftarrow$  g.Players.begin()
- last  $\leftarrow$  g.Players.end()
- fonction  $\leftarrow$  suivant quel critère  $A < B$ ?

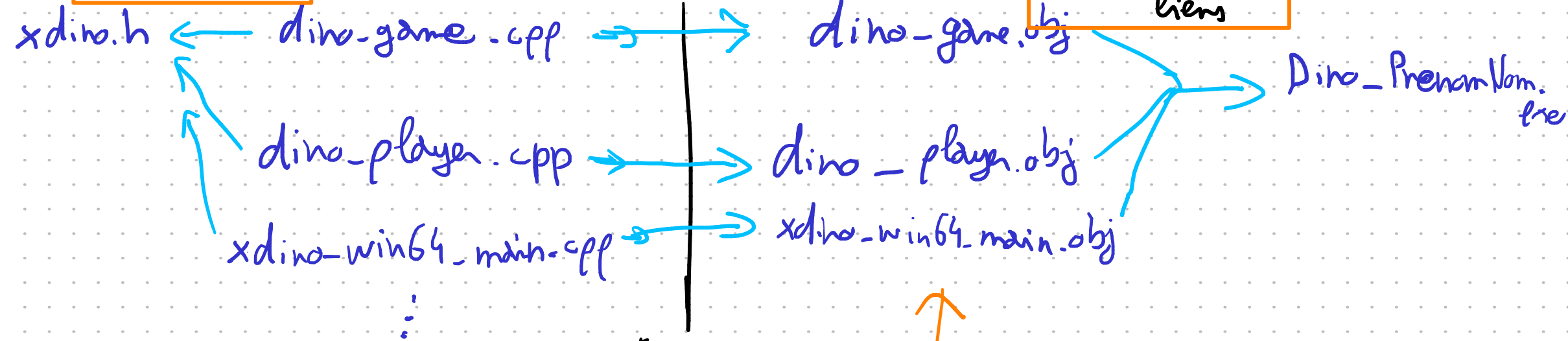
```
bool DinoPlayer::IsAbove(DinoPlayer & p2) {  
    return m_pos.y < p2.m_pos.y  
}
```

```
bool ComparePlayersPos(DinoPlayer & a, DinoPlayer & b) {  
    return a.IsAbove(b);  
}
```

1/Préprocesseur CPP  
#include

2/Compilation

3/Édition des liens EXE



UNITÉS DE  
COMPILATION

Instructions assembleur pour le CPU  
\* Dépend de la plateforme  
\* Dépend de la config (options ou pas)

