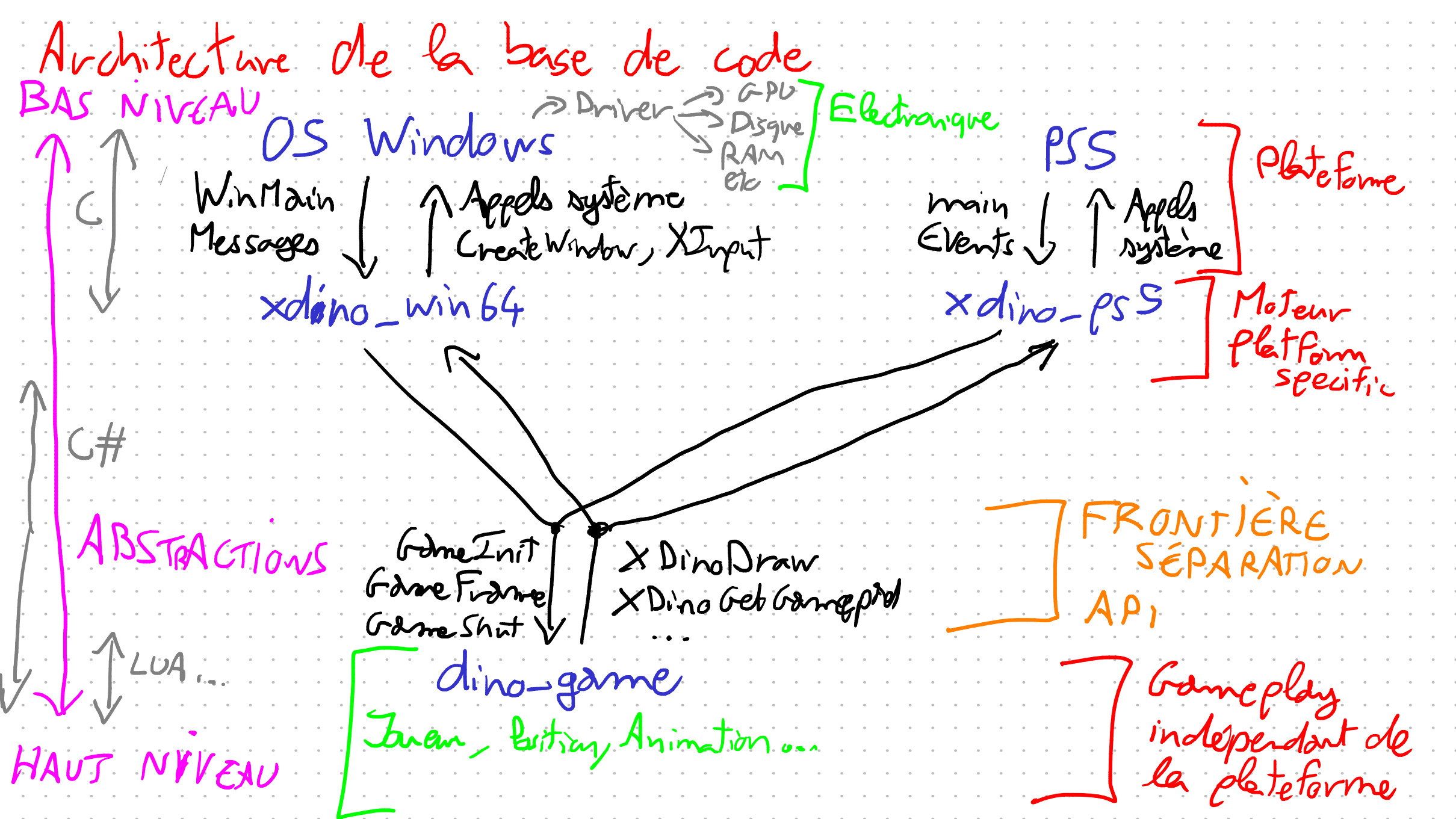
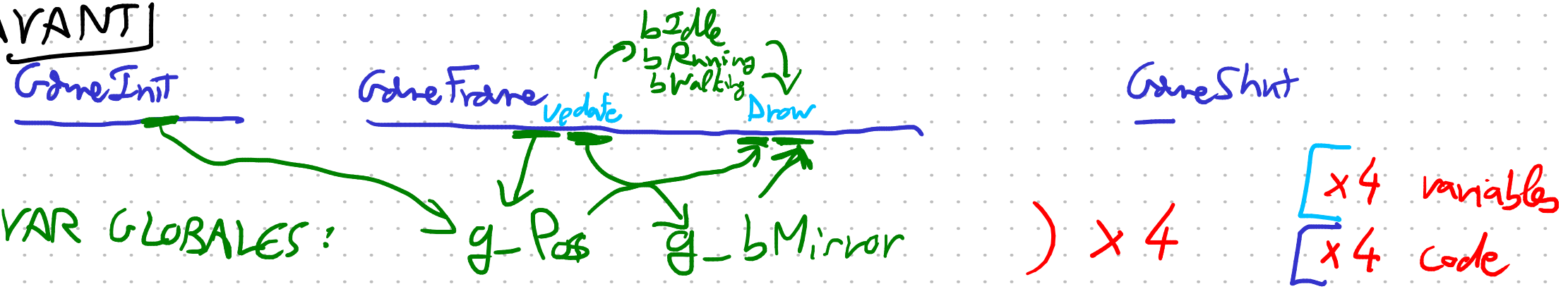
Architecture de la base de code



AVANT



APRÈS

```
Struct DinoPlayer {
    pos
    bMirror
    bIdle
    bRunning
    bWalking
    void UpdatePlayer();
    void DrawPlayer();
}; DinoPlayer p1, p2, p3, p4;
```

```
void DinoPlayer::UpdatePlayer(  )
void DinoPlayer::DrawPlayer(  )

player1.UpdatePlayer(—)
    this -> pos        ou    pos
    this -> bMirror          bMirror
```

CE QU'ON A FAIT

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

MEMBRE — IMPLICITE

] FONCTIONS MEMBRES = MÉTHODES

CE QU'ON AURAIT PU FAIRE

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

EXPLICITE

(Par pointeur pour ne pas copier)

] FONCTIONS 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);
private:
    m_pos;  m_bMirror
    m_bWalking  m_bRunning
                m_bIdle
    m_color
};
```

Public = comportement
Private = implémentation