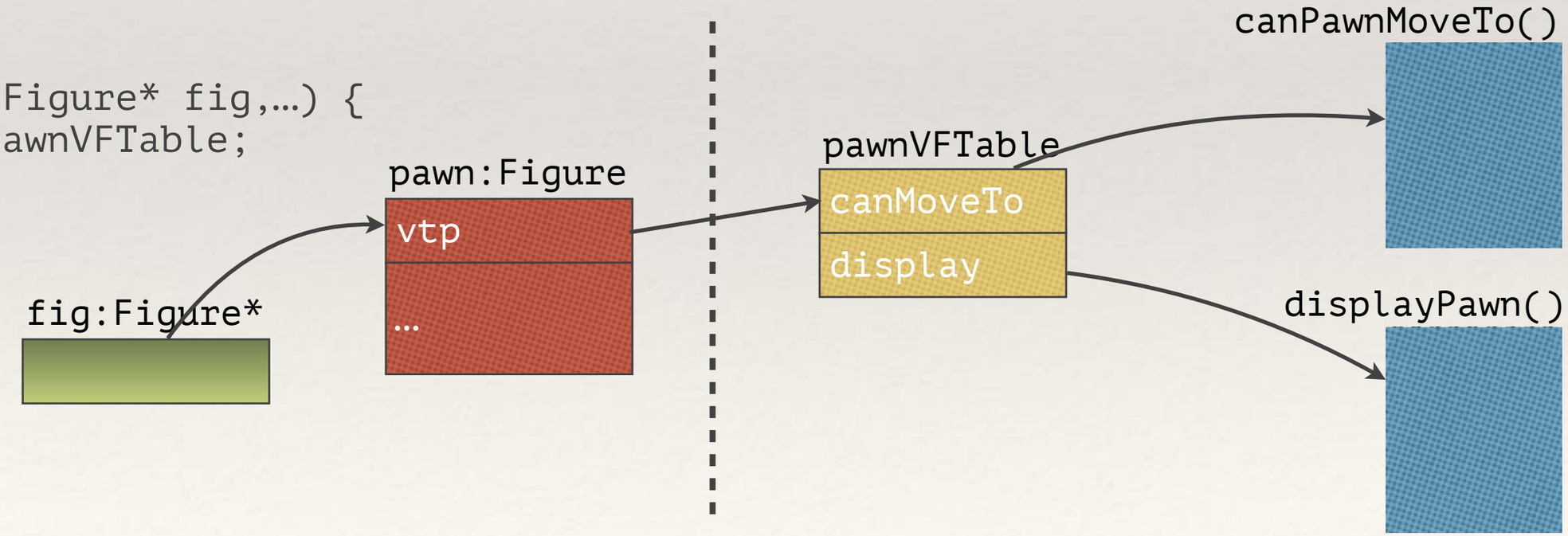
Sa proceduralnog na OO programiranje: polimorfizam

- Kod za funkcije generiše prevodilac, a prikazane strukture (tabele) postoje po jedna za svaku klasu (vrstu objekta) i inicijalizuju se statički, za vreme prevođenja
- Svaki objekat ima pokazivač na takvu tabelu pokazivača na implementacije virtuelnih funkcija koje odgovaraju svakoj pojedinoj klasi (vrsti objekta), tzv. *virtual table pointer*
- Ovaj pokazivač potrebno je inicijalizovati za svaki objekat, u zavisnosti od njegove vrste:

```
struct Figure {
  Figure_VFTable* vtp;  // Virtual table pointer
  FigureColor color;
  …
};

void initPawn (Figure* fig,…) {
  fig->vtp = &pawnVFTable;
  …
}
```



Sa proceduralnog na OO programiranje: polimorfizam

- Sada se *polimorfan* poziv realizuje jednostavno, *dinamičkim vezivanjem* (*dynamic binding*), preko lanca pokazivača, uvek istim kodom:

```
int canMoveTo (Figure* this, unsigned col, unsigned row) {
  return this->vtp->canMoveTo(this,col,row);
}

int display (Figure* this, …) {
  return this->vtp->display(this,…);
}
```

- Pogodnost: kod pozivaoca se nimalo ne menja proširenjem hijerarhije klasa (dodavanjem novih podvrsta)