Flyweight, proxy, object pool

Łukasz Milewski

Uniwersytet Wrocławski

May 6, 2010, Wrocław

Pytania?

Tematy do dyskusji

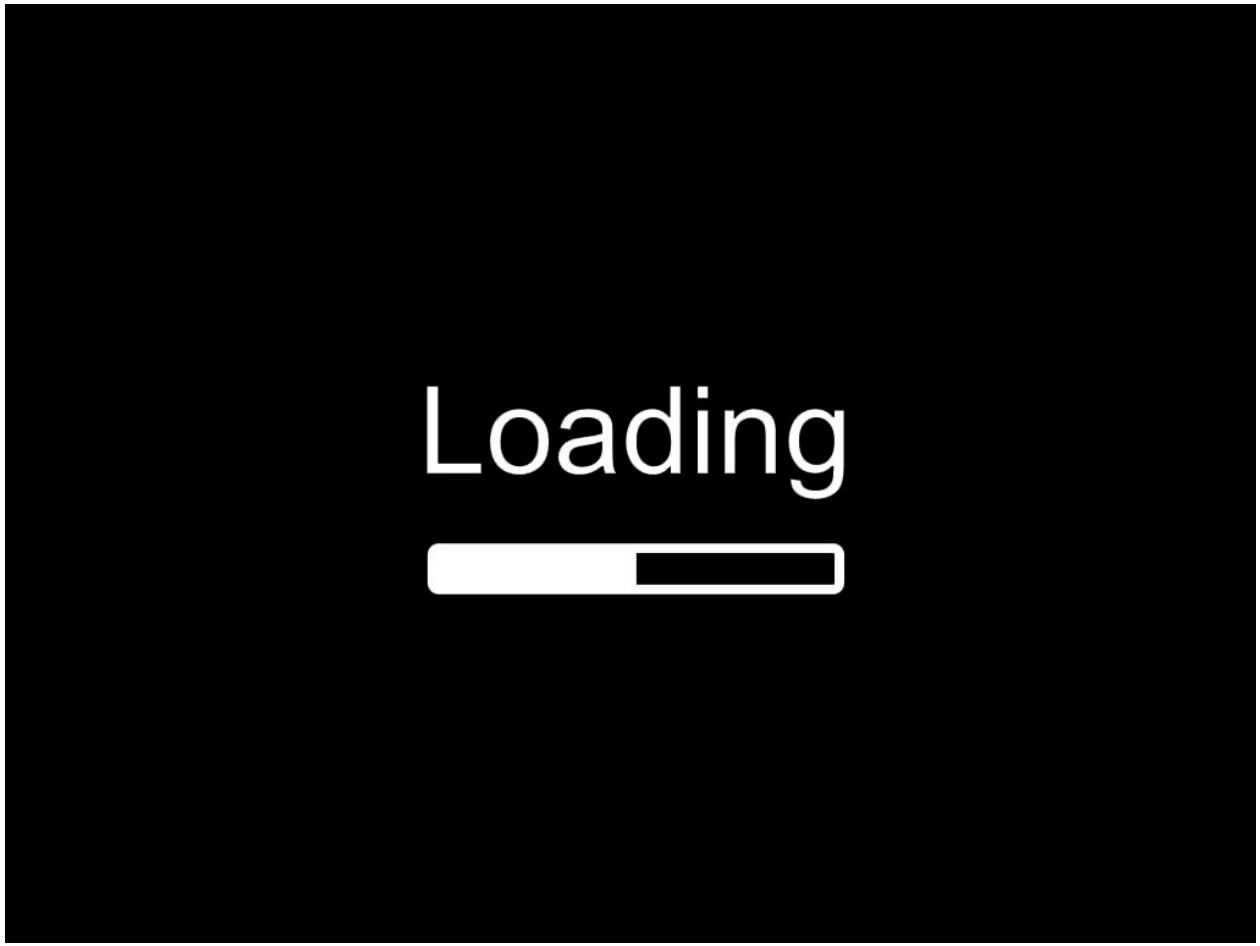
Co chcemy wiedzieć

- Może praca w grupie?

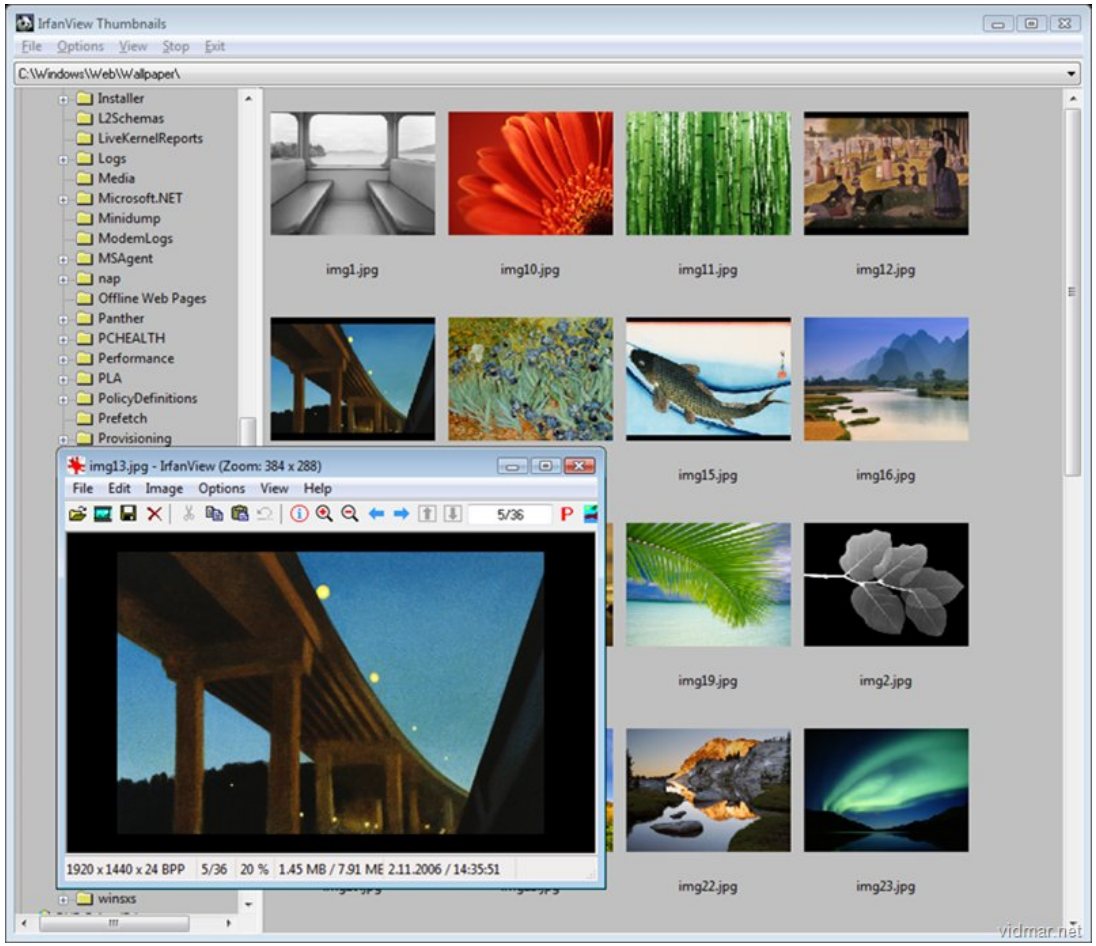
Ciekawostka

Ciekawostka

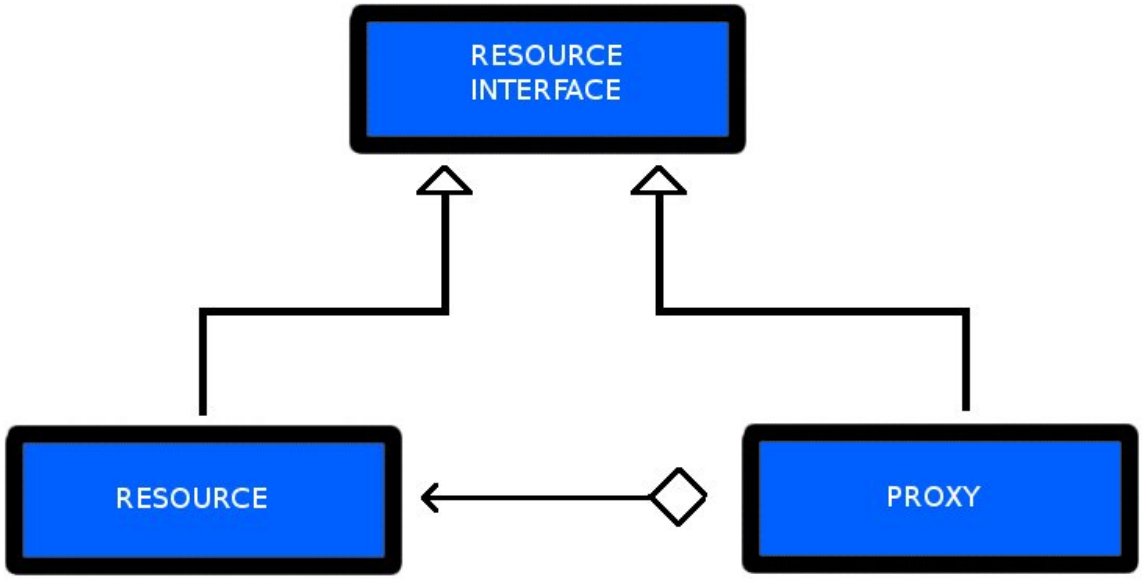
Nasz problem



Nasz problem



Rozwiązanie



Praktyka

Zastosowania

- Remote proxy
- Virtual proxy
- Protection proxy
- Smart reference
- COW

Koszt

Co tracimy?

- Nieznany koszt operacji
- Koszt wywołania wirtualnego (Protection proxy w OSie?)
- Dodatkowy kod

Pytania?

A może jednak pytania?

Tematy do dyskusji

- Jakie jeszcze widzimy zastosowania dla wzorca proxy?
- W jaki sposób proxy może zmniejszyć elastyczność naszego programu?
- Czym różni się wzorzec proxy od wzorca decorator?
- Czym różni się wzorzec proxy od wzorca adapter?

Nasz problem

Rozwiązanie

Co można poprawić

Zły pomysł

- Na każdy sezon budować karuzelę od nowa
- Po rozbierać karuzelę i wyrzucać ją na złom

Dobry pomysł

- Wybudujmy 'domek' dla naszej karuzeli
- Na początku sezonu ją rozstawiamy
- Po sezonie ją składamy

Analiza

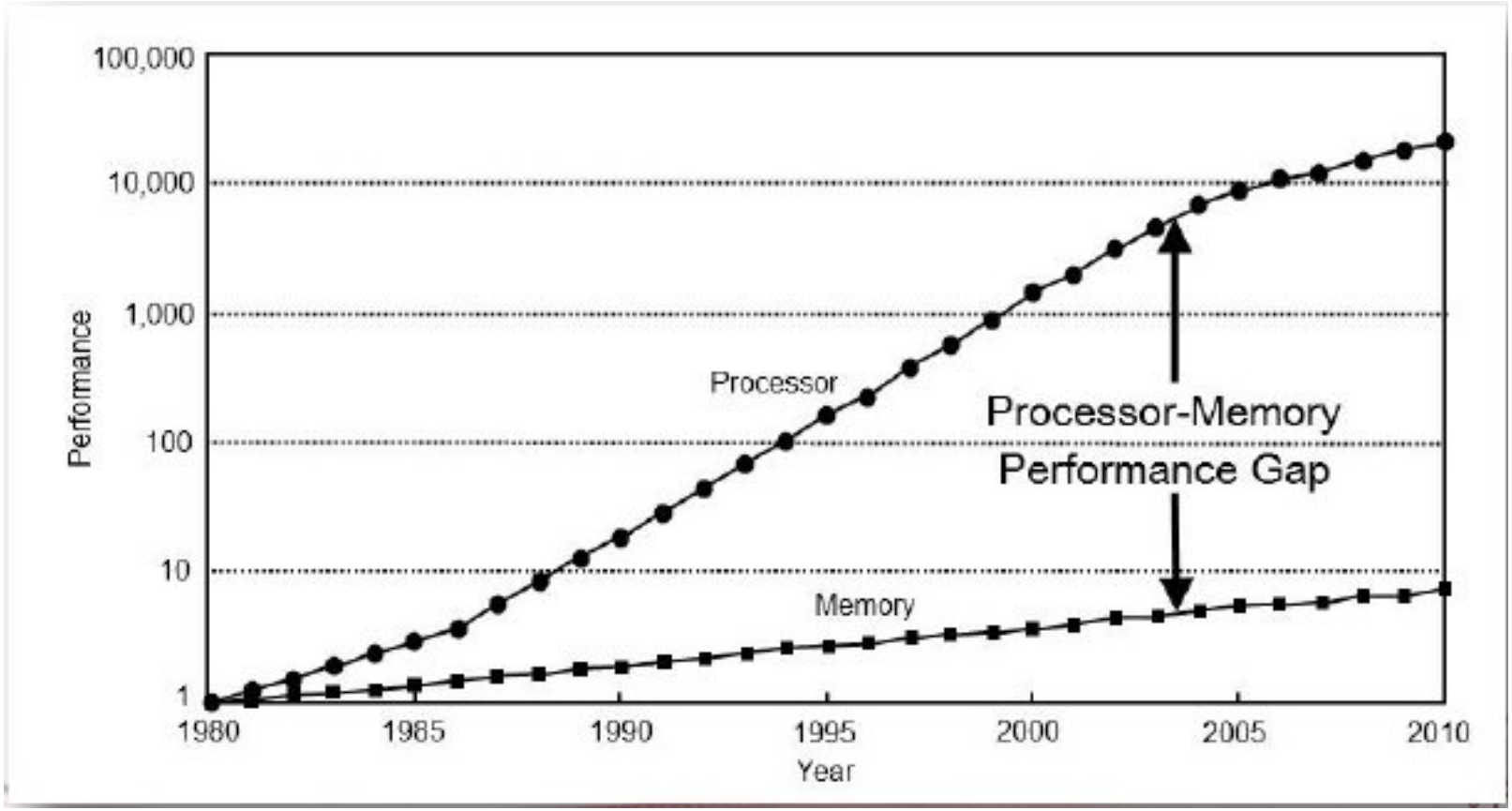
Jeżeli:

- Tworzenie obiektu danego typu jest kosztowne
- Niszczenie obiektu dnageo typu jest kosztowne
- W trakcie działania aplikacji tworzymy dużo obiektów tego typu
- Zazwyczaj nasze obiekty nie żyją zbyt długo

to możemy:

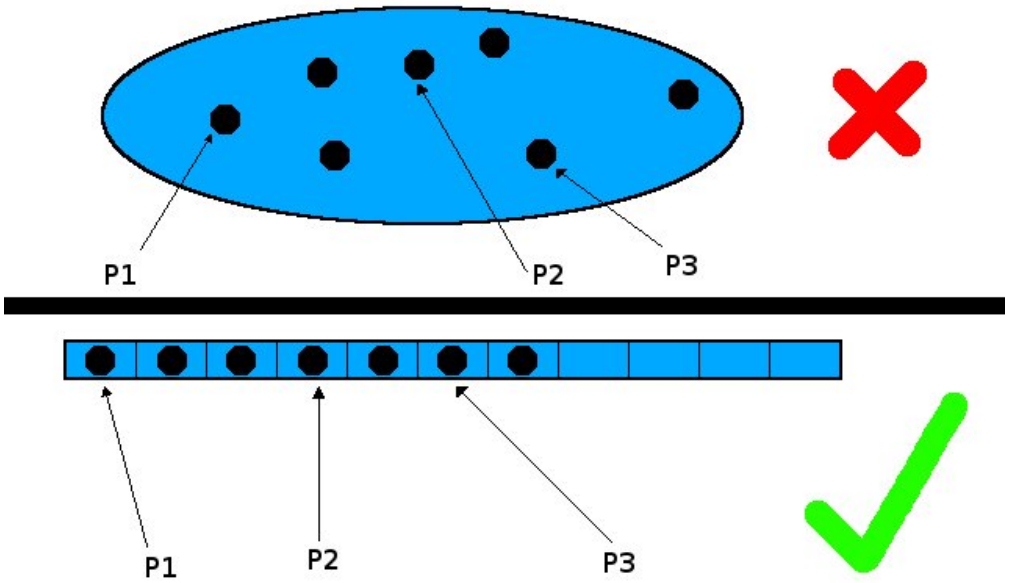
- Zapamiętać N obiektów w 'puli'
- Zamiast tworzyć obiekt - pobieramy go z puli
- Zamiast niszczyć obiekt - odkładamy go do puli

Dodatkowy, ukryty zysk!



// wykres z "Pitfalls of Object Oriented Programming" by Tony Albrecht

Dodatkowy, ukryty zysk!



Co zyskujemy?

Co tracimy?

- Błędy (cesspool)
- Bezpieczeństwo
- Dodatkowa synchronizacja przy wielowątkowości

Pytania?

Tematy do dyskusji

Co chcemy wiedzieć

- W jaki sposób object pool wpływa na elastyczność designu?
- Jak ma się object pool do obecności GC?
- Co zrobić z sytuacją gdy programista zapomni zwolnić zasób?

Nasz problem

Piszemy sobie kompilator, aż tu nagle...

- Lekser
- Obiekty
- Sprawdzanie właściwości znaków / pretty printing
- Uruchamiamy i działa poprawnie, ale bardzo wolno

Co zbudowaliśmy?

Wszędzie obiekty

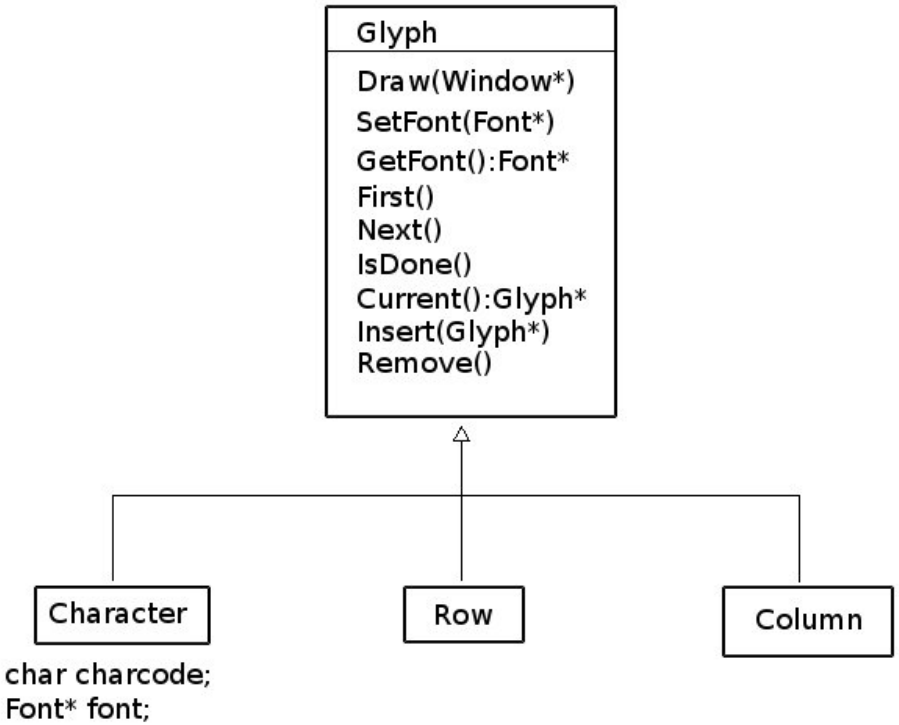
- Przetwarzamy ogromne ilości bardzo małych obiektów
- Wiele z nich musimy pamiętać w trakcie działania programu
- Chcemy jednak zachować elastyczność modelu obiektowego
- Jesteśmy 'tru obdżekt guru' - co powiedzą nasi znajomi?

Weźmy prostszy przykład!

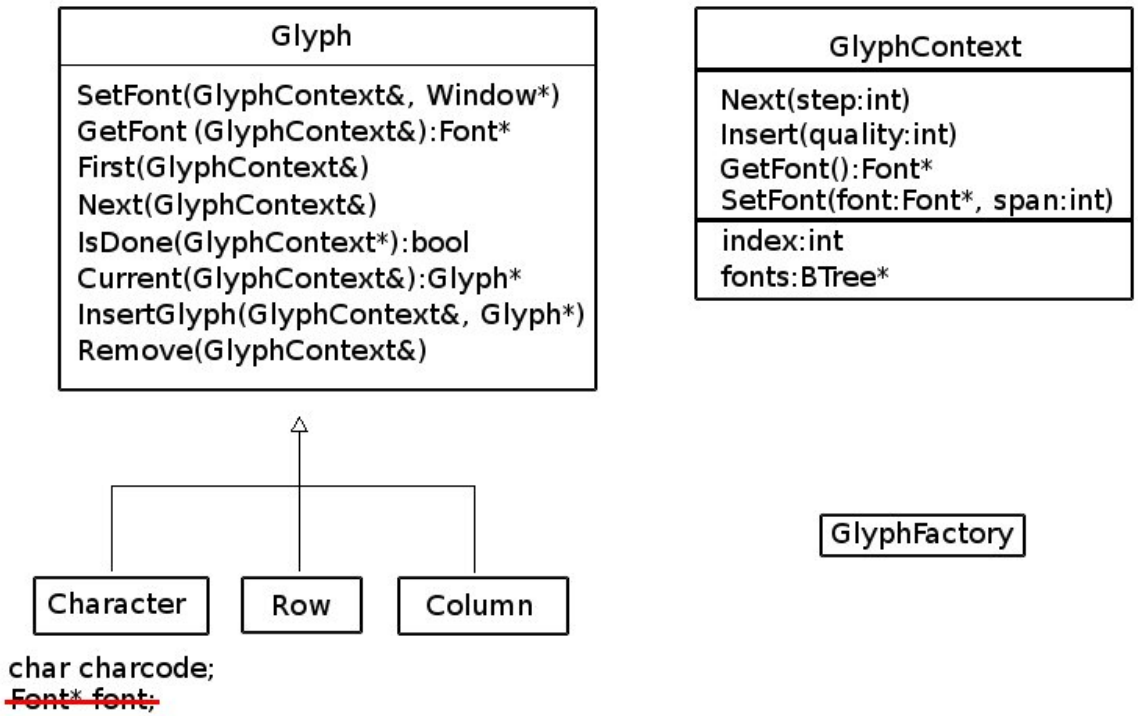
Przykład z obiektem na każdy znak w lekserze jest 'trochę' mało realny.

A może edytor tekstu?

Zły pomysł


Glyph
Draw(Window*)
SetFont(Font*)
GetFont():Font*
First()
Next()
IsDone()
Current():Glyph*
Insert(Glyph*)
Remove()
Character
Row
Column
char charcode;
Font* font;

Flyweight!


Glyph
SetFont(GlyphContext&, Window*)
GetFont (GlyphContext&):Font*
First(GlyphContext&)
Next(GlyphContext&)
IsDone(GlyphContext*):bool
Current(GlyphContext&):Glyph*
InsertGlyph(GlyphContext&, Glyph*)
Remove(GlyphContext&)
GlyphContext
Next(step:int)
Insert(quality:int)
GetFont():Font*
SetFont(font:Font*, span:int)
index:int
fonts:BTree*
Character
Row
Column
GlyphFactory
char charcode;
Font* font;

Zysk

- Elastyczny, obiektowy design
- Dobra wydajność

Koszt

- Konieczność przekazywania kontekstu i dbania o jego poprawność
- Trochę więcej pisania

Pytania?

Tematy do dyskusji

Co chcemy wiedzieć

- Jakie są zastosowania flyweight, z którymi spotykamy się codziennie?
- Jak ma się flyweight do innych wzorców projektowych?


KONIEC