# Use case’y do kompilatora poleceń traktora:

Parser z wklepanych zdań przez użytkownika tworzy obiekt Phrase, który składa się tylko i wyłącznie z listy obiektów TaskCommand – czyli de facto pojedynczych zdań, zawierających jedno polecenie do wykonania i opisu tego polecenia. Kompilator ma działać tak, że dostaje obiekt Phrase i na jego podstawie wykonuje konkretne metody z api gry. Opis TaskCommand to tak naprawdę:

* lista cech czasownika (okolicznik) – czyli np. „natychmiast”, „następnie” itd.
* lista dopełnień (rzeczowniki) na jakich ma zostać wykonana dana czynność, np. „pole”, „pola”, „magazyn” itd. (może ich być wiele)
  + zrozumiałe, że trzeba to przetłumaczyć na wywołanie konkretnej metody dla każdego obiektu odpowiadającemu dopełnieniu
* każde dopełnienie może mieć swoją osobną listę cech (przydawki), np. „(pole) kukurydzy”, „wszystkie niezaorane (pola)”
  + określają konkretne obiekty dla których ma zostać wykonana czynność, tutaj będzie trzeba stworzyć mechanizm filtrowania, np. dla pól tak jak mówiłem wygodnie może być utworzenie jednej listy obiektów, składających się z pól i współrzędnych , z której będzie się zwracać zbiory spełniające wymagania używając LINQ
* parser rozkłada zdanie złożone jakie otrzymuje na wejściu na listę zdań prostych: może być taka sytuacja, że w jednym zdaniu będzie dopełnienie np. „pole” a w drugim wystąpi dopełnienie domyślne, które należy pobrać z pierwszego zdania, np.:
  + „Zaoraj pole kukurydzy i zasiej je” –„je” będzie sparsowane jako pole domyślne
  + „Zasiej wszystkie niezasiane pola i nawodnij wszystkie zasiane” – w podzdaniu „nawodnij wszystkie zasiane” jest podmiot domyślny

tu ważna sprawa – jak na razie na planszy są tylko pola, ale pojawi się magazyn, może sklep może coś (no, prawdopodobnie), te rzeczy będą pewnie zaimplementowane jako osobne klasy i zrobi się rodzica/interfejs wspólny dla nich i pól (tych dotychczasowych); potem w state managerze będzie tablica z tym rodzicem/interfejsem, a nie tak jak do tej pory z polem – dobrze byłoby to napisać tak, żeby w razie czego nie trzeba było przebudowywać wszystkie totalnie - ale generalnie luz, bo jak nam się nie będzie chciało to się po prostu zrobi magazyn jako klasa pochodna dla pola, i metoda typu „zaoraj magazyn” będzie zaimplementowana jako pusta (albo rzuci wyjątek)

fajnie by było więc, gdyby kompilator miał jakąś szczątkową analizę semantyczną, że np. nie można „zaorać magazynu” :D

W dalszej perspektywie jest też do napisania context manager, który będzie wtrącał swoje trzy grosze przy kompilowaniu (np. będzie miał dane o stanie gry i w przypadku zdania „zaoraj to pole” albo „Zacznij zasiewać” zobaczy, że chodzi o pole, które było orzeczeniem w poprzednim zdaniu, pole które jest jako jedyne z otaczających go niezaorane/niezasiane etc), fajnie by było napisać kompilator tak, żebym potem można było dodać tę opcję bez rozwalania tego co jest

# Kilka konkretnych przypadków:

User wklepuje „Zaoraj wszystkie niezaorane i zgniłe pola.”. Parser dzieli wg części zdania, kompilator dostaje phrase, które składa się z jednego TaskCommand (bo jest jedno orzeczenie(polecenie)).

Po czynności dobiera odpowiednią metodę, po dopełnieniu i jego określeniach buduje odpowiedni filtr (tak żeby faktycznie wywołać metodę dla wszystkich niezaoranych lub zgniłych pól (tu jest trochę hardkor z polskim językiem, bo „i” i „lub” robią niezły burdel, ale wyjebane w to na razie)), następnie dla wszystkich przefiltrowanych wyników wywołuje odpowiednią metodę.

„Zaoraj i zasiej wszystkie niezaorane pola,a następnie nawodnij zasiane”.

Tutaj parser ogarnie, że „zaoraj” i „zasiej” dotyczy tych samych pól, a „nawodnij” innych. Dostaniesz phrase, składające się z trzech TaskCommand. W pierwszym orzeczeniem będzie „zaoraj”, w drugim „zasiej” w trzecim „nawodnij”. Określenia dla dwóch pierwszych będą identyczne, czyli taki sam filtr, ale inne metody do wykonania. W trzecim zdaniu jest podmiot domyślny, trzeba ogarnąć że chodzi o „pola”. Pojawia się też okolicznik „a następnie”, teoretycznie powinno to wpływać na kolejkowanie (możesz to zaimplementować, ale nie trzeba) – bo tak czy siak wszystko leci od lewej do prawej.

„Natychmiast zbierz plony z pól kukurydzy”

Tutaj pojawia się okolicznik „natychmiast”, czyli trzeba wydać komendę „stop” traktorowi i dodać dopiero resztę.

Na razie tyle, jak mi się coś przypomni to dam znać jakoś.