Design Pattern for Undo Engine

Asked 16 years, 3 months ago Modified 3 years, 4 months ago

Viewed 40k times

128

I'm writing a structural modeling tool for a civil enginering application. I have one huge model class representing the entire building, which include collections of nodes, line elements, loads, etc. which are also custom classes.

I have already coded an undo engine which saves a deep-copy after each modification to the model. Now I started thinking if I could have coded differently. Instead of saving the deep-copies, I could perhaps save a list of each modifier action with a corresponding reverse modifier. So that I could apply the reverse modifiers to the current model to undo, or the modifiers to redo.

I can imagine how you would carry out simple commands that change object properties, etc. But how about complex commands? Like inserting new node objects to the model and adding some line objects which keep references to the new nodes.

How would one go about implementing that?

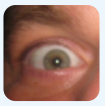
design-patterns **undo**

edited Jul 22, 2014 at 10:32

Improve this question

Follow

asked Sep 8, 2008 at 13:58



Ozgur Ozcitak

10.6k ● 8 ● 49 ● 58

If I add the comment "Undo Algorthim" will that make it so I can search "Undo Algorithm" and find this? That's what I searched for and I found something closed as a duplicate.
– Peter Turner Feb 17, 2009 at 20:43

hay,I also want to develope undo/redo in the application we are developing.We use QT4 framework and need to have many complex undo/redo actions..I was wondering , have you succeed using Command-Pattern ?
– Ashika Umanga Umagiliya Jul 7, 2010 at 3:45

3 @umanga: It worked but it wasn't easy. The hardest part was keeping track of references. For example, when a Frame object is deleted, its child objects: Nodes, Loads acting on it and many other user assignments needed to be kept to be reinserted when undone. But some of these child objects were shared with other objects, and undo/redo logic became quite complex. If the model wasn't that large, I would keep the memento approach; it is much easier to implement.
– Ozgur Ozcitak Oct 6, 2010 at 8:25

this is a fun problem to work on, think about how source code repos do it, like svn (they keep the diffs between commits).
– Alex Aug 22, 2012 at 18:51

22 Answers

Sorted by: Highest score (default)

95

Most examples I've seen use a variant of the Command-Pattern for this. Every user-action that's undoable gets its own command instance with all the information to execute the action and roll it back. You can then maintain a list of all the commands that have been executed and you can roll them back one by one.

Share Improve this answer

Follow

edited Mar 29, 2012 at 0:07

E-rich

9,501 ● 11 ● 49 ● 83

answered Sep 8, 2008 at 14:00

Mendelt

37.5k ● 6 ● 75 ● 97

6 This is basically how the undo engine in Cocoa, NSUndoManager, works. – amrox Sep 8, 2008 at 20:39

What would you say is appropriate when you have some commands which should be undoable and others which shouldn't? In particular when you have an undo/redo manager that is keeping a stack of commands? Perhaps non-undoable commands get their own class, or perhaps their `send-to-undo-manager` method just does nothing? – Eric Auld Mar 16, 2021 at 20:22

1 @EricAuld I think how you implement that depends a whole lot on what your application is actually doing. Subclassing commands sounds like a good idea anyway. Not just for undoable and un-undoable commands but for different types of commands. But like I said, that depends a lot on the implementation. – Mendelt Mar 26, 2021 at 8:53

I think both memento and command are not practical when you are dealing with a model of the size and scope that the OP implies. They would work, but it would be a lot of work to maintain and extend.

For this type of problem, I think you need to build in support to your data model to support differential checkpoints for *every object* involved in the model. I've done this once and it worked very slick. The biggest thing you have to do is avoid the direct use of pointers or references in the model.

Every reference to another object uses some identifier (like an integer). Whenever the object is needed, you lookup the current definition of the object from a table. The table contains a linked list for each object that contains all the previous versions, along with information regarding which checkpoint they were active for.

Implementing undo/redo is simple: Do your action and establish a new checkpoint; rollback all object versions to the previous checkpoint.

It takes some discipline in the code, but has many advantages: you don't need deep copies since you are doing differential storage of the model state; you can scope the amount of memory you want to use (*very* important for things like CAD models) by either number of redos or memory used; very scalable and low-maintenance for the functions that operate on the model

since they don't need to do anything to implement undo/redo.

Share Improve this answer

Follow

answered Feb 17, 2009 at 21:09

Jeff Kotula

2,134 ● 12 ● 17

1 If you use a database (eg sqlite) as your file format this can be almost automatic – Martin Beckett Sep 8, 2009 at 2:00

4 If you augment this by tracking dependencies introduced by changes to the model, then you could potentially have an undo tree system (i.e. if I change the width of a girder, then go do some work on a separate component, I can come back and undo the girder changes without losing the other stuff). The UI for that might be a little unwieldy but it would be much more powerful than a traditional linear undo. – Sumudu Fernando Apr 29, 2012 at 6:22

Can you explain this id's vs pointers idea more? Surely a pointer/memory address works just as well as id? – paulm May 12, 2014 at 12:49

1 @paulm: essentially the actual data is indexed by (id, version). Pointers refer to a particular version of an object, but you're seeking to refer to the current state of an object, whatever that may be, so you want to address it by id, not by (id, version). You *could* restructure it so that you store a pointer to the (version => data) table and just pick the latest each time, but that tends to harm locality when you're persisting data, muddies concerns a little, and makes it harder to do some sorts of common queries, so it's not the way it would normally be done. – Chris Morgan Jul 7, 2020 at 16:22

If you're talking GoF, the Memento pattern specifically addresses undo.

17

Share Improve this answer

Follow

answered Sep 8, 2008 at 15:00

Andy Whitfield

2,403 ● 2 ● 20 ● 22

9 Not really, this addresses his initial approach. He's asking for an alternative approach. The initial being storing the full state for each step while the latter being storing only the "diffs". – Andrei Rînea Nov 12, 2010 at 1:05

17

As others have stated, the command pattern is a very powerful method of implementing Undo/Redo. But there is important advantage I would like to mention to the command pattern.

When implementing undo/redo using the command pattern, you can avoid large amounts of duplicated code by abstracting (to a degree) the operations performed on the data and utilize those operations in the undo/redo system. For example in a text editor cut and paste are complementary commands (aside from the management of the clipboard). In other words, the undo operation for a cut is paste and the undo operation for a paste is cut. This applies to much simpler operations as typing and deleting text.

The key here is that you can use your undo/redo system as the primary command system for your editor. Instead of writing the system such as "create undo object, modify the document" you can "create undo object, execute redo operation on undo object to modify the document".

Now, admittedly, many people are thinking to themselves "Well duh, isn't part of the point of the command pattern?" Yes, but I've seen too many command systems that have two sets of commands, one for immediate operations and another set for undo/redo. I'm not saying that there won't be commands that are specific to immediate operations and undo/redo, but reducing the duplication will make the code more maintainable.

Share Improve this answer

Follow

answered Sep 8, 2008 at 17:12

Torlack

4,475 ● 1 ● 25 ● 24

2 I've never thought of `paste` as `cut` ^-1. – Lenar Hoyt Nov 15, 2013 at 21:41

8

You might want to refer to the Paint.NET code for their undo - they've got a really nice undo system. It's probably a bit simpler than what you'll need, but it might give you some ideas and guidelines.

-Adam

Share Improve this answer

answered Sep 8, 2008 at 14:05

Follow

Adam Davis
93.5k ● 60 ● 271 ● 333

5 Actually, the Paint.NET code is no longer available, but you can get the forked code.google.com/p/paint-mono
– Igor Brejc Jun 17, 2009 at 12:10

7

I've implemented complex undo systems sucessfully using the Memento pattern - very easy, and has the benefit of naturally providing a Redo framework too. A more subtle benefit is that aggregate actions can be contained within a single Undo too.

In a nutshell, you have two stacks of memento objects. One for Undo, the other for Redo. Every operation creates a new memento, which ideally will be some calls to change the state of your model, document (or whatever). This gets added to the undo stack. When you do an undo operation, in addition to executing the Undo action on the Memento object to change the model back again, you also pop the object off the Undo stack and push it right onto the Redo stack.

How the method to change the state of your document is implemented depends completely on your implementation. If you can simply make an API call (e.g. ChangeColour(r,g,b)), then precede it with a query to get and save the corresponding state. But the pattern will also support making deep copies, memory snapshots, temp file creation etc - it's all up to you as it is is simply a virtual method implementation.

To do aggregate actions (e.g. user Shift-Selects a load of objects to do an operation on, such as delete, rename, change attribute), your code creates a new Undo stack as a single memento, and passes that to the actual operation to add the individual operations to. So your action methods don't need to (a) have a global stack to worry about and (b) can be coded the same whether they are executed in isolation or as part of one aggregate operation.

Many undo systems are in-memory only, but you could persist the undo stack out if you wish, I guess.

Share Improve this answer

Follow

answered Sep 8, 2008 at 17:03

Greg Whitfield

5,719 ● 2 ● 31 ● 32

6

This might be a case where CSLA is applicable. It was designed to provide complex undo support to objects in Windows Forms applications.

Share Improve this answer

Follow

answered Sep 8, 2008 at 14:10

Eric Z Beard

38.4k ● 27 ● 101 ● 147

Just been reading about the command pattern in my agile development book - maybe that's got potential?

5

You can have every command implement the command interface (which has an Execute() method). If you want undo, you can add an Undo method.

more info here

Share Improve this answer

Follow

answered Sep 8, 2008 at 14:02

Dave Arkell

3,980 ● 2 ● 23 ● 28

4

I'm with Mendelt Siebenga on the fact that you should use the Command Pattern. The pattern you used was the Memento Pattern, which can and will become very wasteful over time.

Since you're working on a memory-intensive application, you should be able to specify either how much memory the undo engine is allowed to take up, how many levels of undo are saved or some storage to which they will be persisted. Should you not do this, you will soon face errors resulting from the machine being out of memory.

I would advise you check whether there's a framework that already created a model for undos in the programming language / framework of your choice. It is nice to invent new stuff, but it's better to take something already written, debugged and tested in real scenarios. It would help if you added what you're writing this in, so people can recommend frameworks they know.

Share Improve this answer

edited May 23, 2017 at 12:02

Follow

Community Bot
1 ● 1

answered Sep 8, 2008 at 15:51

Omer van Kloeten
12k ● 9 ● 44 ● 54

4

[Codeplex project]:

It's a simple framework to add Undo/Redo functionality to your applications, based on the classical Command design pattern. It supports merging actions, nested transactions, delayed execution (execution on top-level transaction commit) and possible non-linear undo history (where you can have a choice of multiple actions to redo).

Share Improve this answer

Follow

edited Aug 3, 2021 at 19:11

community wiki
3 revs, 3 users 50%
sg7

3

For reference, here's a simple implementation of the Command pattern for Undo/Redo in C#: [Simple undo/redo system for C#].

Share Improve this answer

Follow

edited Mar 13, 2018 at 0:05

sg7
6,288 ● 2 ● 34 ● 41

answered Mar 19, 2009 at 18:49

Tomas Andrle

13.3k ● 15 ● 79 ● 94

2

Most examples I've read do it by using either the command or memento pattern. But you can do it without design patterns too with a simple deque-structure.

Share Improve this answer

Follow

answered Sep 8, 2008 at 16:08

Patrik Svensson

13.8k ● 8 ● 58 ● 77

What would you put in the deque? – Iraimbilanja Mar 19, 2009 at 19:05

In my case I put the current state of the operations I wanted undo/redo functionality for. By having two deques (undo/redo) i do undo on the undo queue (pop first item) and insert it into the redo dequeue. If the number of items in the dequeues exceed the prefered size i pop an item of the tail.
– Patrik Svensson Mar 25, 2009 at 15:08

2 What you describe actually *IS* a design pattern :). The problem with this approach is when your state takes a lot of memory - keeping several dozens of state version then becomes unpractical or even impossible. – Igor Brejc Jun 17, 2009 at 12:24

Or you can store pair of closure representing normal and undo operation. – Xwtek Oct 3, 2018 at 15:32

2

I had to do this when writing a solver for a peg-jump puzzle game. I made each move a Command object that held enough information that it could be either done or undone. In my case this was as simple as storing the starting position and the direction of each move. I then stored all these objects in a stack so the program could easily undo as many moves as it needed while backtracking.

Share Improve this answer

Follow

edited Dec 11, 2009 at 15:16

answered Sep 8, 2008 at 14:13

Bill the Lizard

405k ● 211 ● 572 ● 889

2

A clever way to handle undo, which would make your software also suitable for multi user collaboration, is implementing an operational transformation of the data structure.

This concept is not very popular but well defined and useful. If the definition looks too abstract to you, this project is a successful example of how an operational transformation for JSON objects is defined and implemented in Javascript

Share Improve this answer

Follow

answered Sep 5, 2014 at 12:09

danza

12.2k ● 8 ● 41 ● 48

1

We reused the file load and save serialization code for “objects” for a convenient form to save and restore the entire state of an object. We push those serialized objects on the undo stack – along with some information about what operation was performed and hints on undo-ing that operation if there isn’t enough info gleaned from the serialized data. Undo and Redoing is often just replacing one object with another (in theory).

There have been many MANY bugs due to pointers (C++) to objects that were never fixed-up as you perform some odd undo redo sequences (those places not updated to safer undo aware “identifiers”). Bugs in this area often ...ummm... interesting.

Some operations can be special cases for speed/resource usage - like sizing things, moving things around.

Multi-selection provides some interesting complications as well. Luckly we already had a grouping concept in the code. Kristopher Johnson comment about sub-items is pretty close to what we do.

Share Improve this answer

Follow

answered Sep 8, 2008 at 16:43

Aardvark

8,561 ● 7 ● 47 ● 64

This sounds increasingly unworkable as the size of your model grows. – Warren P Aug 6, 2010 at 15:15

In what way? This approach keeps working without changes as new "things" are added to each object. Performance could be an issue as the serialized form of the objects grows in size - but this hasn't been a major problem. The system has been under continuous development for 20+ years and is in use by 1000s of users. – Aardvark Aug 9, 2010 at 14:42

1

You can try ready-made implementation of Undo/Redo pattern in PostSharp.
https://www.postsharp.net/model/undo-redo

It lets you add undo/redo functionality to your application without implementing the pattern yourself. It uses Recordable pattern to track the changes in your model and it works with INotifyPropertyChanged pattern which is also implemented in PostSharp.

You are provided with UI controls and you can decide what the name and granularity of each operation will be.

Share Improve this answer Follow

edited May 4, 2016 at 10:55

answered May 4, 2016 at 9:44

Antonín Procházka
1,408 ● 9 ● 17

1

You can make your initial idea performant.

Use persistent data structures, and stick with keeping a list of references to old state around. (But that only really works if operations all data in your state class are immutable, and all operations on it return a new version---but the new version doesn't need to be a deep copy, just replace the changed parts 'copy-on-write'.)

Share Improve this answer

Follow

answered Jul 26, 2016 at 6:36

Matthias

162 ● 2 ● 12

0

I once worked on an application in which all changes made by a command to the application's model (i.e. CDocument... we were using MFC) were persisted at the end of the command by updating fields in an internal database maintained within the model. So we did not have to write separate undo/redo code for each action. The undo stack simply remembered the primary keys, field names and old values every time a record was changed (at the end of each command).

Share Improve this answer

Follow

answered Sep 8, 2008 at 15:17

Agnel Kurian

59.4k ● 47 ● 153 ● 225

The first section of Design Patterns (GoF, 1994) has a use case for implementing the undo/redo as a design

0

pattern.

Share Improve this answer

Follow

answered Feb 17, 2009 at 20:46

Peter Turner

11.4k ● 10 ● 70 ● 113

0

I've found the Command pattern to be very useful here. Instead of implementing several reverse commands, I'm using rollback with delayed execution on a second instance of my API.

This approach seems reasonable if you want low implementation effort and easy maintainability (and can afford the extra memory for the 2nd instance).

See here for an example: https://github.com/thilo20/Undo/

Share Improve this answer

Follow

answered Mar 12, 2018 at 23:24

Thilo

69 ● 1 ● 4

-1

I don't know if this is going to be of any use to you, but when I had to do something similar on one of my projects, I ended up downloading UndoEngine from http://www.undomadeeasy.com - a wonderful engine and I really didn't care too much about what was under the bonnet - it just worked.

Share Improve this answer

Follow

answered Sep 28, 2010 at 15:56

NativeBreed

15

Please post your comments as answer only if you are confident to provide solutions! Otherwise prefer to post it as comment under the question! (if it doesn't allow to do so now! please wait till you get good reputation)
– Rookie Programmer Aravind Dec 11, 2012 at 6:46

-1

In my opinion, the UNDO/REDO could be implemented in 2 ways broadly. 1. Command Level (called command level Undo/Redo) 2. Document level (called global Undo/Redo)

Command level: As many answers point out, this is efficiently achieved using Memento pattern. If the command also supports journalizing the action, a redo is easily supported.

Limitation: Once the scope of the command is out, the undo/redo is impossible, which leads to document level(global) undo/redo

I guess your case would fit into the global undo/redo since it is suitable for a model which involves a lot of memory space. Also, this is suitable to selectively undo/redo also. There are two primitive types

1. All memory undo/redo
2. Object level Undo Redo

In "All memory Undo/Redo", the entire memory is treated as a connected data (such as a tree, or a list or a graph) and the memory is managed by the application rather than the OS. So new and delete operators if in C++ are overloaded to contain more specific structures to effectively implement operations such as a. If any node is modified, b. holding and clearing data etc., The way it functions is basically to copy the entire memory(assuming that memory allocation is already optimized and managed by the application using advanced algorithms) and store it in a stack. If the copy of the memory is requested, the tree structure is copied based on the need to have a shallow or deep copy. A deep copy is made only for that variable which is modified. Since every variable is allocated using custom allocation, the application has the final say when to delete it if need be. Things become very interesting if we have to partition the Undo/Redo when it so happens that we need to programatically-selectively Undo/Redo a set of operation. In this case, only those new variables, or deleted variables or modified variables are given a flag so that Undo/Redo only undoes/redoes those memory Things become even more interesting if we need to do a partial Undo/Redo inside an object.

When such is the case, a newer idea of "Visitor pattern" is used. It is called "Object Level Undo/redo"

2. Object level Undo/Redo: When the notification to undo/redo is called, every object implements a streaming operation wherein, the streamer gets from the object the old data/new data which is programmed. The data which is not be disturbed is left undisturbed. Every object gets a streamer as argument and inside the UNDo/Redo call, it streams/unstreams the data of the object.

Both 1 and 2 could have methods such as 1. BeforeUndo() 2. AfterUndo() 3. BeforeRedo() 4. AfterRedo(). These methods have to be published in the basic Undo/redo Command (not the contextual command) so that all objects implement these methods too to get specific action.

A good strategy is to create a hybrid of 1 and 2. The beauty is that these methods(1&2) themselves use command patterns

answered May 6, 2016 at 13:29

Parthasarathy SRINIVASAN

21 ● 3