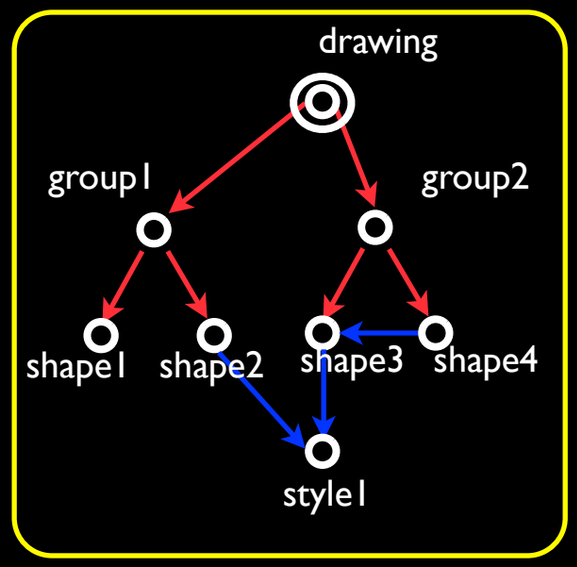
history tracks

- Root embedded object
- Embedded object
- Persistent root
- Composite reference
- Reference
- Inter-persistent-root reference
- Root object to copy
- History track
- Current state
- Current state (computed)

suppose we try to implement history tracks which expose an undo/redo api for a subset of the embedded objects.

In this example let's do history tracks for group1 and group2. (including those objects referenced by composite refs.)

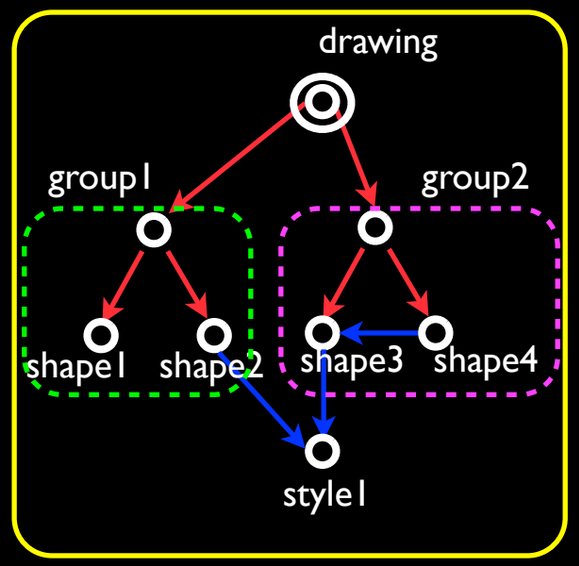


Root embedded object
Embedded object
Persistent root
Composite reference
Reference
Inter-persistent-root reference
Root object to copy
History track
Current state
Current state (computed)

history tracks

persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3



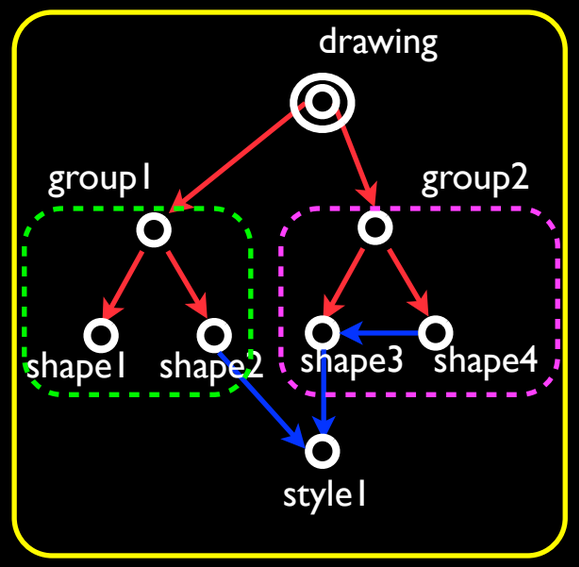
Suppose the user performs the list of edits in the table on the right.

Root embedded object
Embedded object
Persistent root
Composite reference
Reference
Inter-persistent-root reference
Root object to copy
History track
Current state
Current state (computed)

Suppose the user performs the list of edits in the table on the right.

Below, the group1 track and group2 track tables show the edits grouped by history track

history tracks



persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3

group1 track

1	Edit shape1
3	Edit shape2

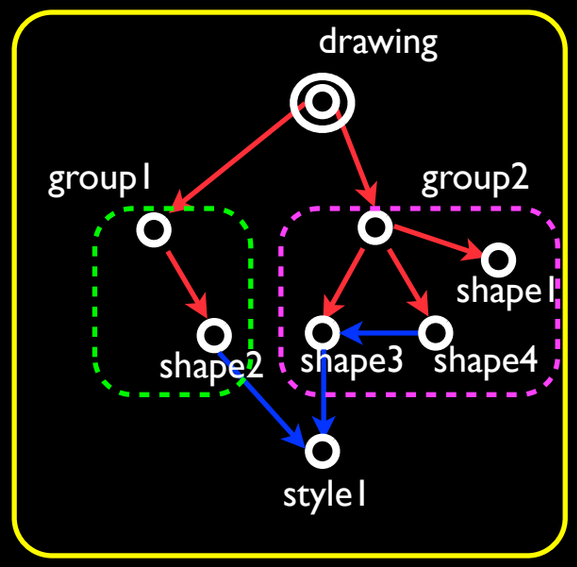
group2 track

2	Edit shape4
4	Edit shape3

- Root embedded object
- Embedded object
- Persistent root
- Composite reference
- Reference
- Inter-persistent-root reference
- Root object to copy
- History track
- Current state
- Current state (computed)

One more edit added here...

history tracks



persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3
5	Move shape1 from group1 to group2

group1 track

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 track

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2

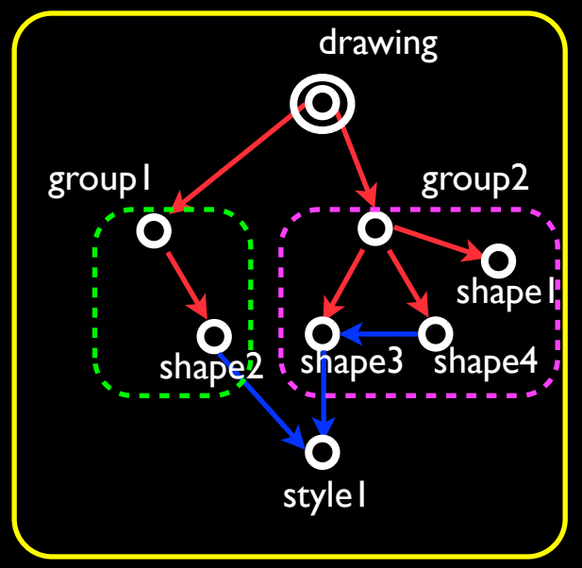
- Root embedded object
- Embedded object
- Persistent root
- Composite reference
- Reference
- Inter-persistent-root reference
- Root object to copy
- History track
- Current state
- Current state (computed)

It's not clear whether the list of changes in each history track should be persistent or if it should be computed at runtime. Suppose we compute it

This means also computing the current state for the tracks from the current state of the persistent root.

This should work fine if we ignore undo/redo; i.e. the tracks are read-only. The track states are computed by scanning backwards through the persistent root history, starting from the persistent root current state.

history tracks: computed



persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3
5	Move shape1 from group1 to group2

group1 track (computed from context history)

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 track (computed from context history)

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2

- Root embedded object
- Embedded object
- Persistent root
- Composite reference
- Reference
- Inter-persistent-root reference
- Root object to copy
- History track
- Current state
- Current state (computed)

Computed: Problem #1
Suppose we want to undo the latest change on track1, "Delete shape1 from group1."

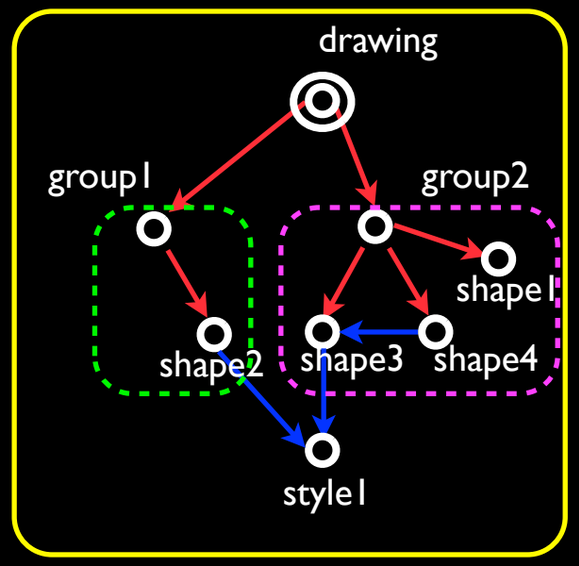
This involves making a commit to the persistent root's history.

We can't undo "Delete shape1 from group1" on its own because that would restore shape1, but there is already an existing shape1.

The only way to perform the undo on track1 is to also undo the last change on track2.

This is somewhat of a problem because it defeats the whole purpose of history tracks (being able to undo/redo changes in subsets of a persistent root, *independently of other tracks*). My conclusion is, if we want to be able to undo/redo independently on history tracks, embedded objects can't be shared between them. However, we can't enforce that in the store if history tracks are just computed at runtime.

history tracks: computed



persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3
5	Move shape1 from group1 to group2

group1 track (computed from context history)

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 track (computed from context history)

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2

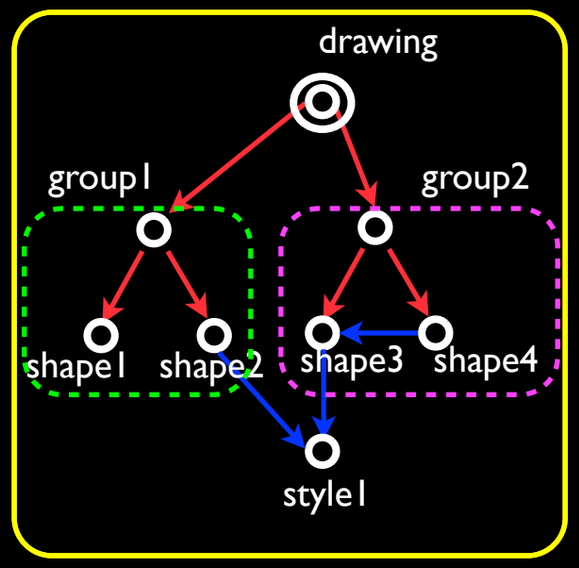
- Root embedded object
- Embedded object
- Persistent root
- Composite reference
- Reference
- Inter-persistent-root reference
- Root object to copy
- History track
- Current state
- Current state (computed)

Computed: Problem #2

Once we make a commit which undoes the change in group1 and group2, it's not clear how to compute the new state group1 and group2 tracks. Remembering that only the "persistent root history" is persistent, how would we compute that the next undo on "group1 track" is implemented by reverting commit #3?

I can imagine an algorithm that scans backwards through the persistent root history, using a stack to keep track of what edits have been undone or redone, but it would not be simple.

history tracks: computed



persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3
5	Move shape1 from group1 to group2
6	Undo Move shape1 from group1 to group2

group1 track (computed from context history)

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 track (computed from context history)

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2

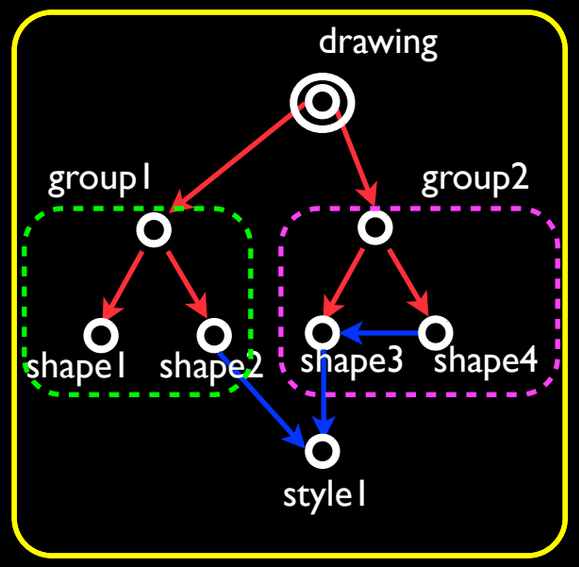
Root embedded object
Embedded object
Persistent root
Composite reference
Reference
Inter-persistent-root reference
Root object to copy
History track
Current state
Current state (computed)

Computed: Problem #3

If the purpose of history tracks is to expose a undo/redo API which would be connected directly to undo/redo UI actions, one problem is that undo/redos performed on tracks create *regular commits* in the persistent root history. this means that if undo/redo for the entire persistent root is also hooked up to UI undo/redo actions, sometimes they will undo/redo regular changes, but sometimes they will undo/redo the *undo/redo actions performed on tracks*.

I think this will be really confusing. Cmd+Z should never undo *an undo performed earlier by Cmd+Z*.

history tracks: computed



persistent root
history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3
5	Move shape1 from group1 to group2
6	Undo Move shape1 from group1 to group2

group1 track
(computed from context history)

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 track
(computed from context history)

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2

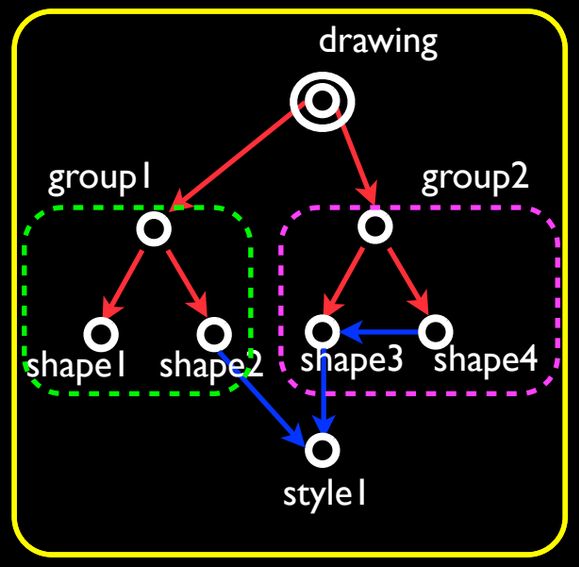
- Root embedded object
- Embedded object
- Persistent root
- Composite reference
- Reference
- Inter-persistent-root reference
- Root object to copy
- History track
- Current state
- Current state (computed)

history tracks: computed

Problem 2 is probably solvable, but it's complex.

Problems 1 and 3 are not solvable without moving to a different implementation of history tracks.

My conclusion from these problems is, if we want the functionality of history tracks, we need to use real persistent roots.



persistent root history: (persistent)

1	Edit shape1
2	Edit shape4
3	Edit shape2
4	Edit shape3
5	Move shape1 from group1 to group2
6	Undo Move shape1 from group1 to group2

group1 track (computed from context history)

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 track (computed from context history)

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2

Root embedded object
Embedded object
Persistent root
Composite reference
Reference
Inter-persistent-root reference
Root object to copy
History track
Current state
Current state (computed)

history tracks: implemented using persistent roots

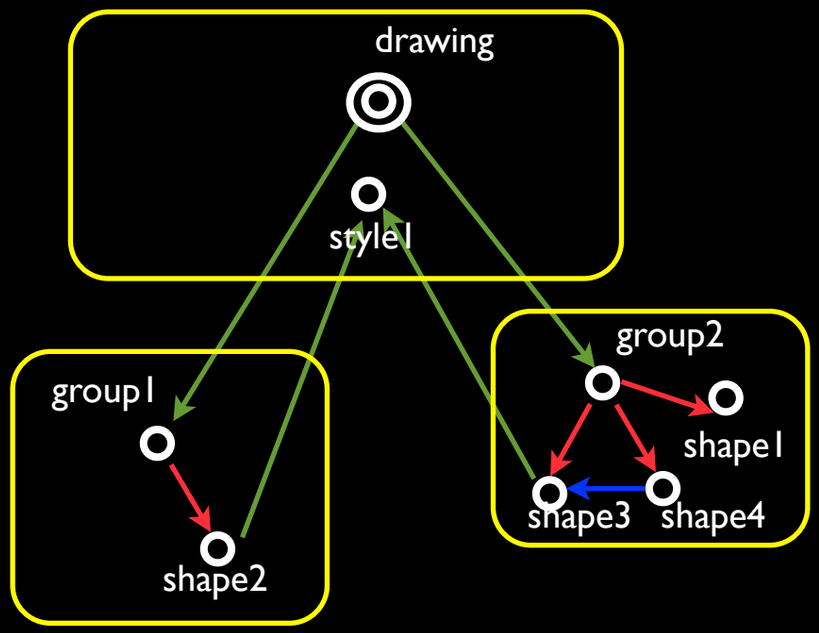
Splitting the original persistent root into several smaller ones gives us the independent undo/redo we want.

However, there are some consequences (not necessarily negative)

- more persistent roots for the user to be aware of

- now that the persistent roots have separate histories, it's harder to treat them as a single unit (tag, branch, etc)

- each persistent root takes on "document semantics" - i.e. when undoing the move of shape1 from group1, group2 is not affected.



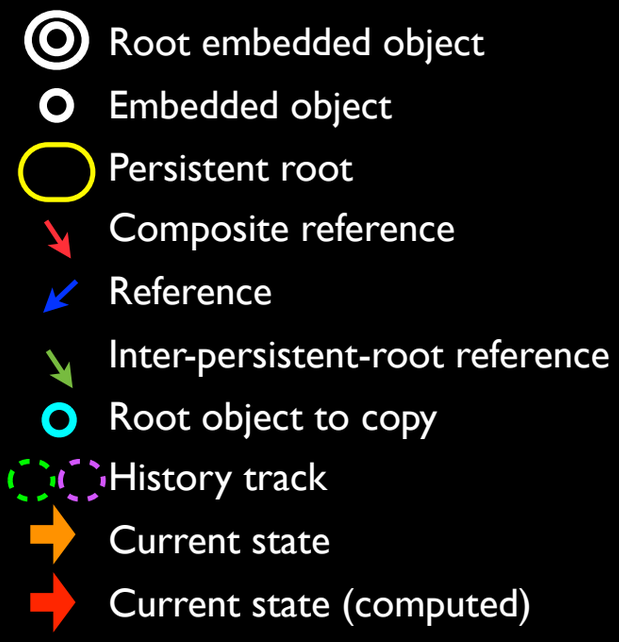
persistent root
history: (persistent)

group1 persistent root
(persistent)

1	Edit shape1
3	Edit shape2
5	Delete shape1 from group1

group2 persistent root
(persistent)

2	Edit shape4
4	Edit shape3
5	Add shape1 to group2



Conclusion

In the end, I don't think history tracks make sense.
Within a persistent root, presenting controls to do
linear undo/redo on a subset of the objects is not practical. This makes sense,
given that the whole motivation for introducing persistent roots was to be able to make
isolated changes (including undo/redo) to subsets of objects in a store.