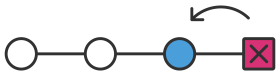
git 실행 취소



이 섹션에서는 사용 가능한 '실행 취소'Git 전략과 명령에 대해 설명합니다. Git은 워드 프로세싱 응용 프로그램에서 발견되는 것과 같은 전통적인 '실행 취소'시스템이 없다는 점에 유의해야합니다. Git 작업을 전통적인 '실행 취소'정신 모델에 매핑하는 것을 자제하는 것이 좋습니다. 또한 Git에는 '실행 취소'작업에 대한 고유 한 명명 규칙이 있으므로 이를 활용하는 것이 가장 좋습니다. 이 용어에는 재설정, 되돌리기, 체크 아웃, 정리 등과 같은 용어가 포함됩니다.

재미있는 은유는 Git을 타임 라인 관리 유틸리티로 생각하는 것입니다. 커밋은 프로젝트 히스토리의 타임 라인을 따라 특정 시점 또는 관심 지점의 스냅 샷입니다. 또한 분기를 사용하여 여러 타임 라인을 관리 할 수 있습니다. Git에서 '실행 취소'할 때, 보통 시간 내에 다시 이동하거나 실수가 발생하지 않은 다른 타임 라인으로 이동합니다.

이 튜토리얼에서는 소프트웨어 프로젝트의 이전 버전을 사용하기 위해 필요한 모든 기술을 제공합니다. 첫째, 이전 커밋을 탐색하는 방법을 보여주는 다음 프로젝트 기록에서 공용 커밋을 되 돌리는 것과 로컬 컴퓨터에서 게시되지 않은 변경을 다시 설정하는 것의 차이점을 설명합니다.

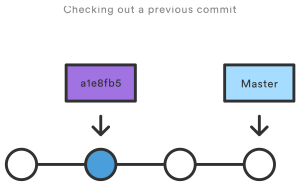
Finding what is lost: Reviewing old commits

모든 버전 제어 시스템의 기본 아이디어는 프로젝트의 "안전한"복사본을 저장하여 코드 기반을 돌이킬 수 없을 정도로 파괴 할 필요가 없다는 것입니다. 커밋의 프로젝트 히스토리를 구축하면 히스토리에서 커밋을 검토하고 다시 방문 할 수 있습니다. Git 저장소의 히스토리를 검토하는 데 가장 좋은 유틸리티 중 하나는 git log 명령입니다. 아래 예제에서 git log를 사용하여 널리 사용되는 오픈 소스 그래픽 라이브러리에 대한 최신 커밋 목록을 가져옵니다.

```
git log --oneline
e2f9a78fe Replaced FlyControls with OrbitControls
d35ce0178 Editor: Shortcuts panel Safari support.
9dbe8d0cf Editor: Sidebar.Controls to Sidebar.Settings.Shortcuts. Clean up.
05c5288fc Merge pull request #12612 from TyLindberg/editor-controls-panel
0d8b6e74b Merge pull request #12805 from harto/patch-1
23b20c22e Merge pull request #12801 from gam0022/improve-raymarching-
example-v2
fe78029f1 Fix typo in documentation
7ce43c448 Merge pull request #12794 from WestLangley/dev-x
17452bb93 Merge pull request #12778 from OndrejSpanel/unitTestFixes
b5c1b5c70 Merge pull request #12799 from dhritzkiv/patch-21
1b48ff4d2 Updated builds.
88adbcdf6 WebVRManager: Clean up.
2720fbb08 Merge pull request #12803 from dmarcos/parentPoseObject
9ed629301 Check parent of poseObject instead of camera
219f3eb13 Update GLTFLoader.js
15f13bb3c Update GLTFLoader.js
6d9c22a3b Update uniforms only when onWindowResize
881b25b58 Update ProjectionMatrix on change aspect
```

각 커밋에는 해시를 식별하는 고유 한 SHA-1이 있습니다. 이러한 ID는 커밋 된 타임 라인을 통해 이동하고 커밋을 다시 방문하는 데 사용됩니다. 기본적으로 git log는 현재 선택된 브랜치에 대해서만 커밋을 보여준다. 찾고자하는 커밋이 다른 지사에 있다는 것은 전적으로 가능합니다. git log --branches = *를 실행하여 모든 브랜치에서 모든 커밋을 볼 수 있습니다. git 브랜치 명령은 다른 브랜치를보고 방문하는 데 사용됩니다. 명령을 호출하면, git branch -a는 모든 알려진 브랜치 이름의 목록을 리턴합니다. git log <branch_name>을 사용하여 이러한 분기 이름 중 하나를 기록 할 수 있습니다.

히스토리에서 방문하고자하는 지점에 대한 커밋 참조를 발견하면 git checkout 명령을 사용하여 커밋을 방문할 수 있습니다. 힘내 체크 아웃은 저장된 스냅 샷을 개발 컴퓨터에 "로드"하는 쉬운 방법입니다. 일반적인 개발 과정에서 HEAD는 대개 마스터 또는 다른 로컬 지사를 가리 킵니다. 그러나 이전 커밋을 체크 아웃 할 때 HEAD는 더 이상 지사를 가리 키지 않습니다. 바로 커밋을 가리 킵니다. 이를 "분리 된 HEAD"상태라고하며 다음과 같이 시각화 할 수 있습니다.



이전 파일을 체크 아웃해도 HEAD 포인터는 이동하지 않습니다. 동일한 분기와 동일한 커밋에 남으며 '분리 된 헤드'상태를 피합니다. 그런 다음 다른 모든 변경 사항과 마찬가지로 새 스냅 샷에서 이전 버전의 파일을 커밋 할 수 있습니다. 결과적으로 파일에서 git checkout을 사용하면 개별 파일의 이전 버전으로 되돌릴 수 있습니다. 이 두 모드에 대한 더 자세한 정보는 git checkout 페이지를 방문하십시오.

과거 커밋 보기

이 예에서는 미친 실험을 시작했다고 가정하지만 계속 유지할지 여부는 확실하지 않습니다. 결정을 돕기 위해 실험을 시작하기 전에 프로젝트의 상태를 살펴보고 싶습니다. 먼저, 보려는 리비전의 ID를 찾아야합니다.

```
git log --oneline
```

프로젝트 내역이 다음과 같다고 가정 해 보겠습니다.

```
b7119f2 Continue doing crazy things
872fa7e Try something crazy
a1e8fb5 Make some important changes to hello.txt
435b61d Create hello.txt
9773e52 Initial import
```

git checkout을 사용하여 다음과 같이 "hello.txt에 대한 일부 변경 사항 가져 오기"커밋을 볼 수 있습니다.

```
git checkout a1e8fb5
```

이렇게하면 작업 디렉토리가 a1e8fb5 커밋의 정확한 상태와 일치하게됩니다. 프로젝트의 현재 상태를 잃지 않고 걱정없이 파일을보고 프로젝트를 컴파일하고 테스트를 실행하고 파일을 편집 할 수 있습니다. 여기서 수행하는 작업은 저장소에 저장되지 않습니다. 계속 개발하려면 프로젝트의 "현재"상태로 돌아 가야합니다.

```
git checkout master
```

여기서는 기본 마스터 분기에서 개발중인 것으로 가정합니다. 일단 master 브랜치로 돌아 왔으면 git revert 나 git reset을 사용하여 원하지 않는 변경을 취소 할 수 있습니다.

Undoing a committed snapshot

기술적으로 커밋을 '실행 취소'하는 여러 가지 전략이 있습니다. 다음 예제는 우리가 다음과 같은 커밋 기록을 가지고 있다고 가정합니다 :

```
git log --oneline
872fa7e Try something crazy
a1e8fb5 Make some important changes to hello.txt
435b61d Create hello.txt
9773e52 Initial import
```

우리는 872fa7e를 실행 취소하는 데 초점을 맞출 것입니다. 어쩌면 일들이 너무 미쳤을 수도 있습니다.

How to undo a commit with git checkout

git checkout 명령을 사용하여 이전 커밋 인 a1e8fb5를 체크 아웃 할 수 있습니다. 미친 커밋이 발생하기 전에 저장소에 상태를 저장합니다. 특정 커밋을 체크 아웃하면 repo가 "분리 된 헤드"상태가됩니다. 즉, 어떤 지점에서도 더 이상 일하지 않습니다. 분리 된 상태에서, 분기를 확립 된 분기로 다시 변경할 때 사용자가 작성하는 모든 새로운 확약은 고아가됩니다. Git의 가비지 컬렉터가 고아가 된 커밋을 삭제합니다. 가비지 수집기는 구성된 간격으로 실행되고 고아 커밋을 영구적으로 파괴합니다. 고아 커밋이 가비지 수집되지 않게하려면 우리가 지점에 있는지 확인해야합니다.

분리 된 HEAD 상태에서 git checkout -b new_branch_without_crazy_commit을 실행할 수 있습니다. 그러면 new_branch_without_crazy_commit이라는 새 분기가 만들어지고 그 상태로 전환됩니다. repo는 이제 872fa7e 커밋이 더 이상 존재하지 않는 새로운 히스토리 타임 라인에 있습니다. 이 시점에서 우리는 872fa7e 커밋이 더 이상 존재하지 않고 '취소 된'것으로 간주하는이 새로운 분기에 대한 작업을 계속할 수 있습니다. 불행히도, 이전 분기가 필요하다면, 아마도 마스터 브랜치 일 것입니다.이 실행 취소 전략은 적절하지 않습니다. 다른 '실행 취소'전략을 살펴 보겠습니다. 더 많은 정보와 예제는 우리의 심도있는 git checkout 토론을 검토한다.

How to undo a public (pushed) commit with git revert

우리가 원래의 커밋 히스토리 예제로 돌아가고 있다고 가정합시다. 872fa7e 커밋을 포함하는 기록. 이번에는 되돌리기 '실행 취소'를 시도해 보겠습니다. git revert HEAD를 실행하면, Git은 마지막 커밋의 역함수로 새로운 커밋을 생성합니다. 이렇게하면 현재 분기 기록에 새 커밋이 추가되고 다음과 같이 표시됩니다.

```
git log --oneline
e2f9a78 Revert "Try something crazy"
872fa7e Try something crazy
a1e8fb5 Make some important changes to hello.txt
435b61d Create hello.txt
9773e52 Initial import
```

이 시점에서 우리는 다시 기술적으로 '취소 된'872fa7 커밋을 수행했습니다. 역사적으로 872fa7e가 존재하지만, 새로운 e2f9a78 커밋은 872fa7e의 변경과 반대입니다. 이전의 결제 전략과 달리 동일한 지점을 계속 사용할 수 있습니다. 이 솔루션은 만족스러운 실행 취소입니다. 이것은 공용 공유 리포지토리를 사용하기위한 이상적인 '실행 취소'방법입니다. 큐레이터와 최소한의 Git 히스토리를 유지해야하는 경우이 전략은 만족스럽지 않을 수 있습니다.

How to undo a commit with git reset

이 실행 취소 전략에 대해서는 작업 예제를 계속 진행합니다. git reset은 여러 용도와 기능을 가진 광범위한 명령입니다. git reset --hard a1e8fb5를 호출하면 커밋 기록이 지정된 커밋으로 재설정됩니다. git log로 커밋 내역을 검사하면 다음과 같이 표시됩니다.

```
git log --oneline
a1e8fb5 Make some important changes to hello.txt
435b61d Create hello.txt
9773e52 Initial import
```

로그 출력은 e2f9a78 및 872fa7e 커밋이 더 이상 커밋 기록에 존재하지 않음을 나타냅니다. 이 시점에서 우리는 작업을 계속하고 마치 "미친"커밋이 발생하지 않은 것처럼 새로운 커밋을 만들 수 있습니다. 변경 사항을 취소하는이 방법은 기록에 가장 명확한 영향을줍니다. 리셋을 수행하는 것은 로컬 변경에 유용하지만 공유 원격 리포지토리로 작업 할 때 문제를 일으 킵니다. 커밋 872fa7e이 푸시된 공유 저장소가 있고, 해당 저장소에 히스토리가 reset 된 브랜치를 `git push` 하려고 시도하면 Git이 이를 잡아 오류를 던질 것이다. Git은 푸시 된 브랜치가 커밋이 누락되어 최신 상태가 아니라고 가정합니다. 이러한 시나리오에서 `git revert` 실행을 취소해야합니다.

Undoing the last commit

이전 섹션에서는 커밋을 취소하기위한 다양한 전략에 대해 논의했습니다. 이러한 전략은 모두 최신 커밋에도 적용됩니다. 그러나 어떤 경우에는 마지막 커밋을 제거하거나 재설정 할 필요가 없을 수도 있습니다. 어쩌면 그것은 조숙하게 만들어 졌을지도 모릅니다. 이 경우 가장 최근의 커밋을 수정할 수 있습니다. git add를 사용하여 작업 디렉토리에서 더 많은 변경을하고 커밋을 위해 준비하면 git commit --amend를 실행할 수 있습니다. 이것은 Git에 구성된 시스템 편집기를 열고 마지막 커밋 메시지를 수정할 수있게합니다. 새로운 변경 사항이 수정 된 커밋에 추가됩니다.

Undoing uncommitted changes

변경 사항이 저장소 히스토리에 커밋되기 전에 스테이징 색인 및 작업 디렉토리에 살고 있습니다. 이 두 영역에서 변경 사항을 실행 취소해야 할 수도 있습니다. 스테이징 색인과 작업 디렉토리는 내부의 Git 상태 관리 메커니즘입니다. 이 두 메커니즘이 어떻게 작동하는지에 대한 자세한 내용을 보려면 git reset 페이지를 방문하십시오.

The working directory

작업 디렉토리는 일반적으로 로컬 파일 시스템과 동기화됩니다. 작업 디렉토리의 변경 사항을 취소하려면 일반적으로 좋아하는 편집기를 사용하는 것처럼 파일을 편집 할 수 있습니다. 힘내는 작업 디렉토리를 관리하는 데 도움이되는 몇 가지 유틸리티를 가지고있다. 작업 디렉토리의 변경을 취소하기위한 편리한 유틸리티 인 git clean 명령이 있습니다. 또한, git reset은 --mixed 또는 --hard 옵션을 사용하여 호출 할 수 있으며 작업 디렉토리에 재설정을 적용합니다.

The staging index

git add 명령은 스테이징 색인에 변경 사항을 추가하는 데 사용됩니다. Git 재설정은 주로 준비 인덱스 변경 사항을 실행 취소하는 데 사용됩니다. - 혼합 리셋은 보류중인 모든 변경 사항을 스테이징 인덱스에서 작업 디렉토리로 다시 이동시킵니다.

Undoing public changes

원격 리포지토리가있는 팀에서 작업 할 때는 변경 사항을 취소 할 때 추가 고려해야합니다. Git 재설정은 일반적으로 '로컬'실행 취소 방법으로 간주되어야합니다. 사설 분기에 대한 변경을 취소 할 때 리셋을 사용해야합니다. 이렇게하면 다른 개발자가 사용하고있는 다른 브랜치에서 커밋을 안전하게 제거 할 수 있습니다. 문제는 공유 분기에서 리셋이 실행될 때 발생하며 그 브랜치는 git push를 사용하여 원격으로 푸시됩니다. 이 시나리오의 경우 Git은 푸시 된 브랜치가 커밋을 잃어 버렸기 때문에 원격 브랜치에 비해 오래된 것으로 인식하여 푸시를 차단할 것이다.

공유 기록을 실행 취소하는 기본 방법은 `git revert` 입니다. 되돌리기는 공유 기록에서 커밋을 제거하지 않기 때문에 재설정보다 안전합니다. 되돌리기는 실행 취소하려는 커밋을 유지하고 원하지 않는 커밋을 반전하는 새로운 커밋을 만듭니다. 이 방법은 공유 된 원격 공동 작업에서 더 안전합니다. 원격 개발자가 분기를 가져 와서 원하지 않는 커밋을 실행 취소하는 새로운 되돌리기 커밋을 수신 할 수 있기 때문입니다.