
www.plasticscm.com
advanced
version
control
POCKET GUIDE
All you need to know about
branching, merging and DVCS
FOR GAME DEVELOPERS
plasticscm

0 introduction

Version control plays a key role in game development, and it is especially relevant for agile teams.

It is the cornerstone for best practices such us continuous integration, continuous delivery and DevOps, and massive art collaboration.

Only when using version control can teams implement the “collective code ownership” and enforce the concept of being “always ready to ship”.

There is one feature that makes all modern version control systems (Git, Mercurial, and Plastic SCM) stand out from the previous generations - they excel on branching and merging, but in addition: Game Development requires managing vast amount of assets and art collaboration too.

The goal of this guide is to be a powerful tool for the expert developer by explaining the key concepts to master the most relevant merging techniques. Mastering branching and merging is the way to master version control.

Plastic Gluon, the new workflow and user interface made for artists and designers, to manage art assets is also introduced.

Finally, references of studios worldwide using Plastic are presented.

Plastic SCM Team – Boecillo Tech Park
https://www.plasticscm.com/company/team.html

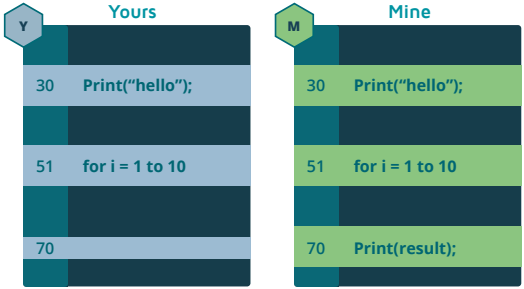
1 2-way merge

Many “arcane” version control systems are only capable of running 2-way merges (SVN, CVS). That’s the reason why most developers fear merging.

All merges are **manual** in a 2-way merge and that’s why they’re slow, boring and error-prone.

Consider the following scenario shown in the picture below: “Did I add the line 70? Or did you delete it?”

There is no way to figure this out!
And this will happen for every single change if you merge using a 2-way merge tool. It is boring for a few files, painful for a few hundreds and simply not doable for thousands.



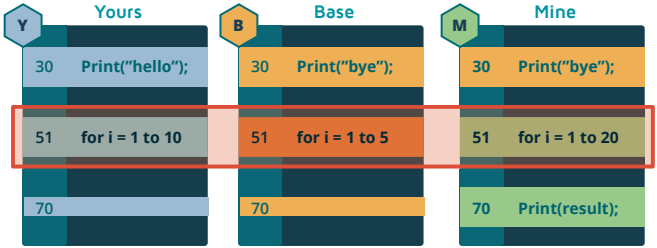
Find out more:
blog.plasticscm.com/2016/02/three-way-merging-look-under-hood.html

2 3-way merge

All modern version control systems (Git, Hg, Plastic SCM) feature improved merge tracking and enable 3-way merging.

3-way merge not only compares “your copy to mine”. It also uses the “base” (a.k.a. common ancestor) to find out “how the code was before our changes”.

This changes everything! Now 99% of merges will be automatic - no manual intervention required.



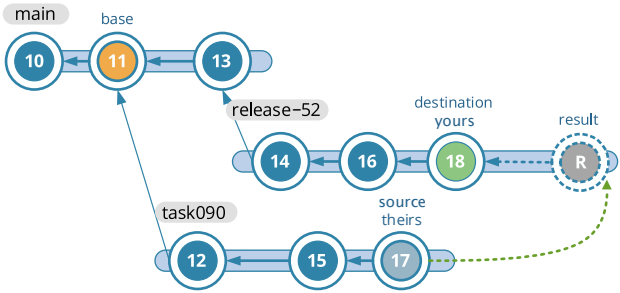
When you compare to the “base”, conflicts are solved this way:

- *Line 30 - automatic – it will just keep yours.*
- *Line 70 - automatic – just keep mine (I added the line).*
- *Line 51 – manual – the user has to decide how to write the “for loop”. Is it from 1 to 10? 1 to 5? 1 to 20? Or maybe write something else?*

Find out more:
blog.plasticscm.com/2016/02/three-way-merging-look-under-hood.html

3 merge contributors

When you merge between two branches, you always deal with the merge contributors:



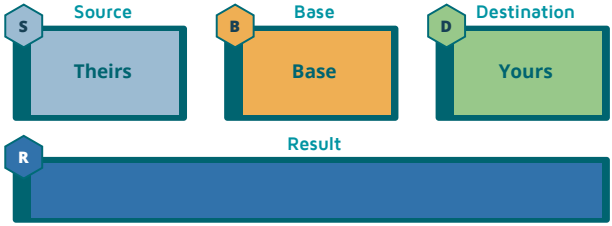
- *The developer needs to merge "17" and "18" and the result of the merge will be placed on branch "release-52".*
- *The version control calculates the "common ancestor" of "17" and "18". In our scenario, the common ancestor, or base, is the changeset "11".*
- *The version control will launch the 3-way merge tool for each file in the conflict. The conflicts will be found comparing "17" and "18" to "11".*

Once the merge is done, the version control will create a "merge link" (the green arrow between "17" and "result") that will be used to calculate the common ancestor in upcoming merges.

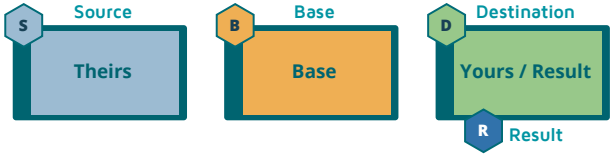
4 merge tool layout patterns

Almost all merge tools (Araxis, Xmerge, BeyondCompare, KDiff3) use one of the following patterns to handle the merge contributors:

- They can use a “4 panel” layout as follows:



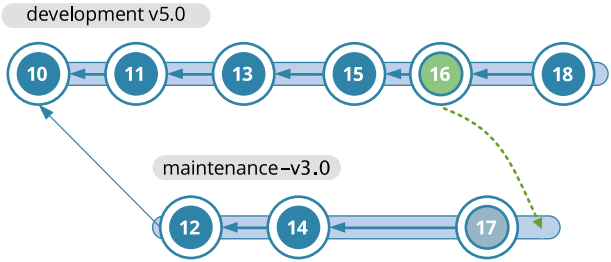
- Or, they can use a “3 panel” layout displaying “yours” and “result” together:



Once you understand this, merge tools won't seem so secretive to you!

5 cherry pick

How can we apply the fix of changeset “16” to the 3.0 branch?



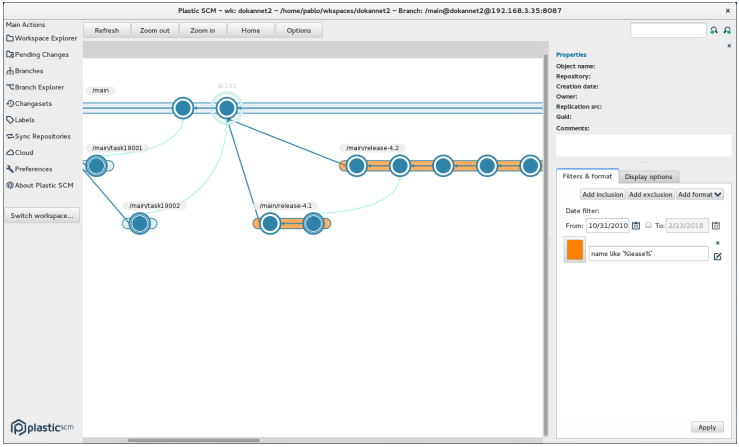
We can’t just merge “16” to “17” because then we’d apply all changes before “16” in branch 5.0 to 3.0. This would basically turn 3.0 into 5.0, which is definitely not what we want.

We just want to apply the “patch” of “16”; the changes made on “16” to the 3.0 branch.

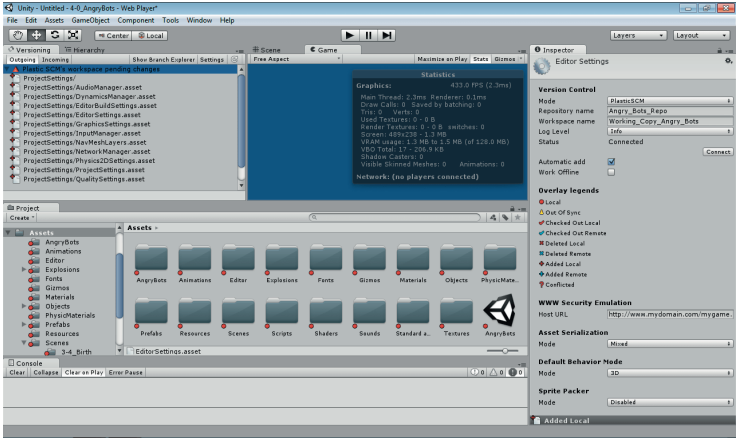
This operation is known as “Cherry Pick”.

Native GUIs

Native Linux and Mac GUIs:

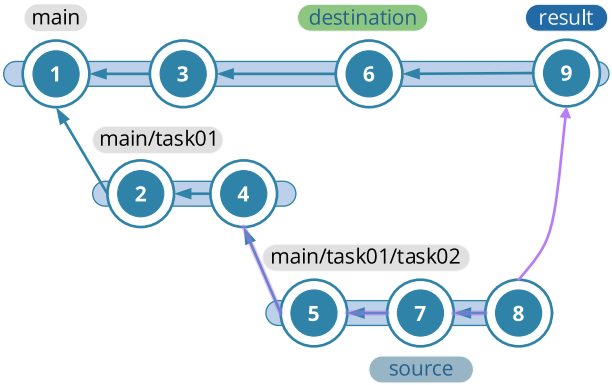


Integration with Unity:



6 branch cherry pick

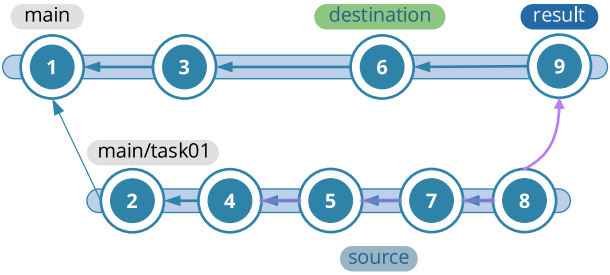
This is just a slightly modified “cherry pick” that allows you to apply a “branch level patch”; it will get the changes made on the branch, but also won't take the parent changes.



The merge in the figure above considers the (4,8] interval; changesets “5”, “7” and “8” will be ‘cherry picked’, but not 2 and 4 as would happen with a regular merge.

7 interval merge

This is yet another way to run a cherry pick. This time the developer selects the beginning and end of the merge interval. This way he chooses exactly what needs to be picked to merge.



The scenario in the figure will get the changes inside the interval (4, 8], which means only “5”, “7” and “8” will be applied.

LEARN HOW STUDIOS WORLDWIDE USE PLASTIC SCM

+350 DEVELOPERS

EPISODIC GAMES

Game of Thrones
Batman
Walking Dead
Minecraft Story Mode

+4TB repos | Artists GUI | Network speed | Tools team supp. | Scalability

+100 DEVELOPERS

Meta 2
Virtual Reality hardware and SDK

Unity | GUI - Branch Expl. | Artists GUI | Huge Repos | Big files | Performance

Find out more:
https://www.plasticscm.com/games/

40 DEVELOPERS

(Fully distributed)

Rust - Survival game

Unity & Unreal | GUI - Branch Expl. | Artists GUI | Huge Repos | Big files | Performance

30 DEVELOPERS

Open world survival game
Subnautica PC and Xbox

Big files | Huge repos | Branching & Merg. | GUI - Branch Expl.

6 DEVELOPERS

Games for healthcare,
education & team building

Unity | +28GB repos | Performance | GUI - Branch Expl. | Branching & Merg.

8 subtractive merge

Subtractive merge is very powerful, but you need to handle it with care.
It is very important to understand that it is not just a "revert". You shouldn't be using subtractives on a regular basis; it is a tool for just special situations.

In the figure above, we need to delete the change done by changeset “92” but keep “93”, “94” and “95”.

We can’t just revert to “92” since we’d lose “93”, “94” and “95”.

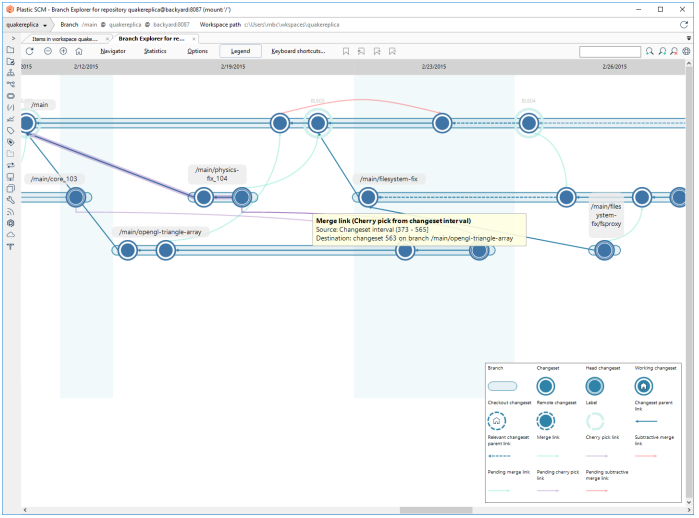
What subtractive does is the following: 96=91–92+93+94+95.

It is an extremely powerful tool to “disintegrate” tasks, but you really need to know what you’re doing.

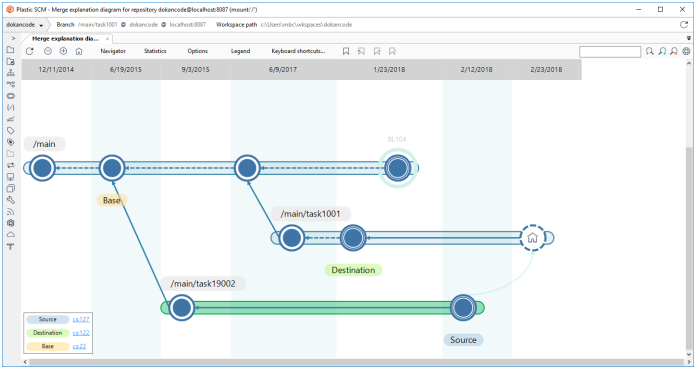
If you want to learn more about branching, merging, and Plastic SCM, take a look at **"the merge machine".**

https://www.plasticscm.com/mergemachine/index.html

The Plastic SCM Branch Explorer rendering different types of merges and its legend:



Plastic SCM includes a feature to "explain merges" and it renders the merge contributors visually, as you can see in the screenshot:



9 DVCS

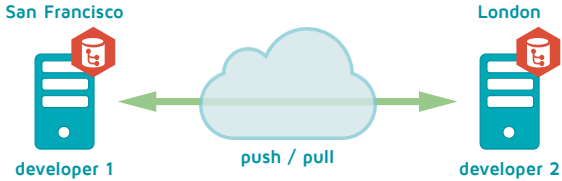
DVCS (distributed version control system) is the concept that took the industry by storm in the last decade.

Thanks to the new breed of tools such as Git, Mercurial, and of course Plastic SCM, version control is no longer considered a commodity and it is now seen as a competitive advantage.

With DVCS, teams don't depend anymore on a single central repository (and central server) since now there can be many clones and changes are "pushed and pulled" among them. In the case of Plastic SCM, there can be even partial clones.

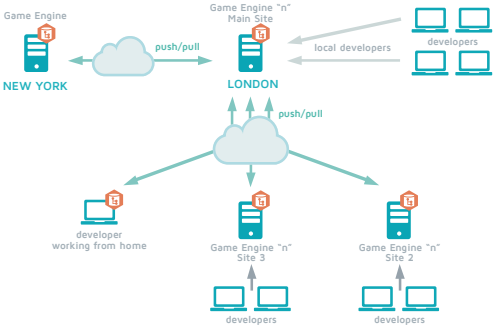
Now teams working away from the head office don't have to suffer slow connections anymore, solving one of the classic issues in globally distributed development.

DVCS brings freedom and flexibility to the design of the repository and the server structure.



Learn more about the history of version control:
https://www.plasticscm.com/version-control-history.html

Distributed teams and studios handled with Plastic SCM



DESCRIPTION

01 The company has different studios around the world.

02 The engine studio is in New York. Different studios will push and pull to it.

03 One of the studios, in London, has different teams that are also distributed in different sites.

04 The scenario shows a hierarchy of servers:

- One 'main engine' server in New York.
- One 'studio' central server in London.
- Developers working in centralized mode in London (direct LAN connection to the server).
- Two additional sites with additional servers. They'll push and pull from the London server.
- One developer working from home and having a local, partial replica of the repository (or repositories) he needs to work.

BENEFITS

01 Multiple servers that can use different underlying technology: suppose the New York studio is a 'Windows shop' while the London one prefers to run servers on Linux. Replication will work independently of the underlying operating system and database backend.

02 Servers running with different DB backends and HW requirements:

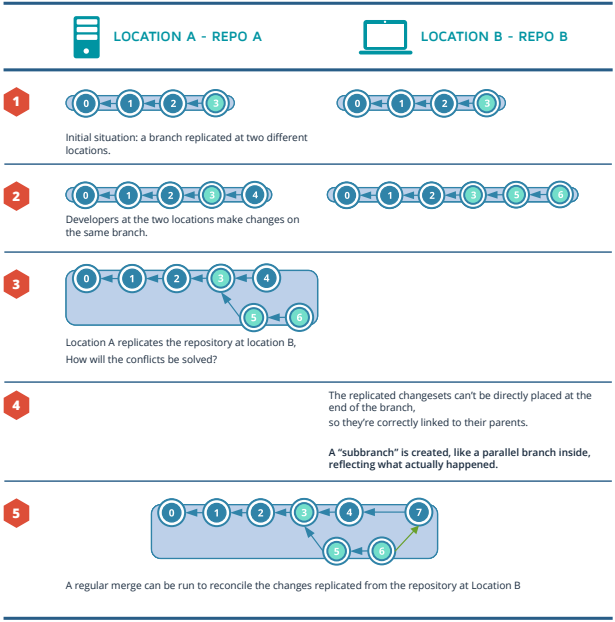
- The New York studio can be a big server (32Gb RAM, 16 cores) using a SQL Server and handling the load of 400 concurrent developers.
- The London studio can use a big server (16Gb, 8 cores) using MySQL and handling the load of 100 concurrent developers.
- The developer working from home can use a replicated server running on his laptop and use a SQLite backend (really small memory footprint and only handling single user load).
- The 2 distributed sites inside the 'UK Studio' can use much smaller machines to handle the load of 10 developers each.

03 Replication between environments ensures proper asset and code delivery.

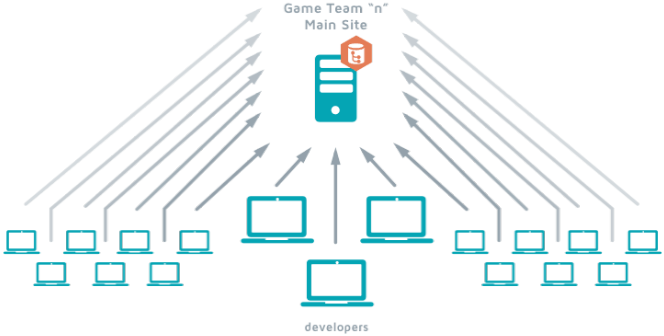
10 solving distributed conflicts

What happens when two developers work on the same branch on different repository clones? How will they reconcile the concurrent changes?

The figure below explains it step by step:



Plastic SCM is a DVCS but supports working centralized



DESCRIPTION

01 Coders and Artists in a given location can work directly connected to the version control server, without a local clone.

02 The direct connection enables a simpler "lock-edit-checkin" cycle which is more natural for artists and coders. Disk space is also saved, considering in game dev projects it is measured in GB.

03 Avoid unwanted locks, be responsive and fast, so the main server seamlessly scales to handle from few to hundreds of concurrent users.

BENEFITS

01 Plastic SCM supports both centralized and distributed scenarios. A workspace can be directly connected to a central server; no need for intermediate repository clones (although you may use them). That's why Plastic SCM can work in "real" centralized mode.

02 Plastic supports heavy load on a single server.

03 Multiple Plastic servers can be configured to split the repository catalog among multiple machines, if necessary.

04 Speed is also remarkably fast, consistently beating established competitors.

Plastic Gluon

Plastic SCM is very visual already and the latest version adds a **new GUI + workflow specifically designed for artists** in game development: Plastic Gluon.

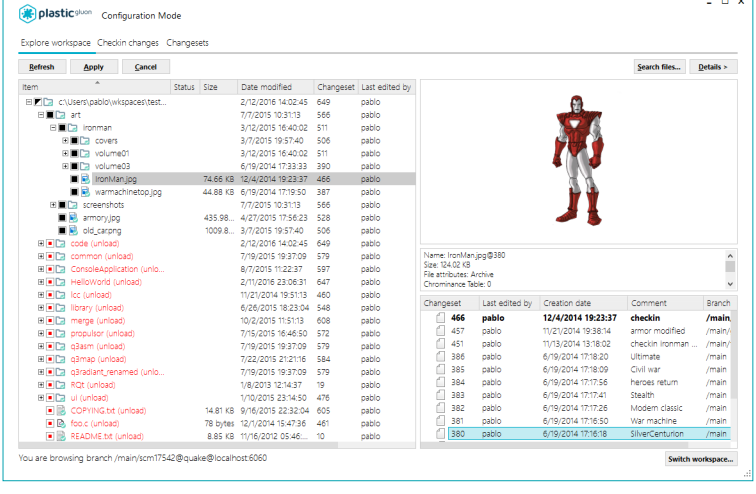
Plastic Gluon removes all the "coder-oriented" bits and keeps what is relevant for an artist.
It is all about an "artist-centric" vision:

01 The artist can select which files to download (file-per-file workflow instead of the more developer-oriented 'changeset mode').

02 Sync only what you need – saving time with huge binaries.

03 Able to checkin even if part of the workspace is out-of-sync (per file version control).

Coders and artists will use a single version control but differently and always taking advantage of what makes them productive.



11 about us

Códice Software was founded back in 2005 with one goal in mind: develop a high performance distributed version control system for really advanced teams.

250 sprints and more than 800 releases later, Plastic SCM helps teams in more than 20 countries build better software. Teams in well-known companies like Microsoft, Samsung, Delphi, Siemens, Medtronic, HP, Mapfre, DHL, Facepunch Studios, and TellTale. They all have something in common: they need the best branching and merging system, high performance, high scalability and distributed development.

Videogame studios around the world are currently switching to Plastic SCM because it is the only DVCS able to handle huge files, locks, simplifies artists workflow and collaboration, and combine centralized and distributed development.

Maybe the code of your favorite videogame is already controlled by Plastic SCM... :-)

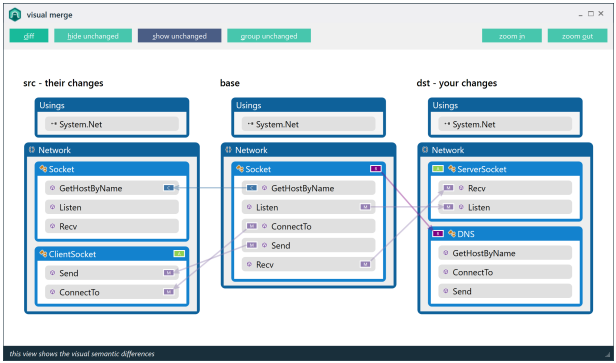
12 about semanticmerge

All our work happens to be around merging source code - Merging files, directories, finding conflicts... creating ways to make merging simpler and more effective.

Programmers used to ask us: “Why merge tools don’t understand the code?”

And that’s why we created SemanticMerge in 2013; **the first 3-way merge tool** in the market **able to understand the code.**

We applied all that we learned with Plastic about merging plus all the ideas we had about how merge should be. SemanticMerge is the first step towards semantic version control.



www.semanticmerge.com

Boecillo Tech Park
Edificio Centro, 103
47151 Valladolid - SPAIN

sales@codicesoftware.com

@plasticscm

PlasticSCM