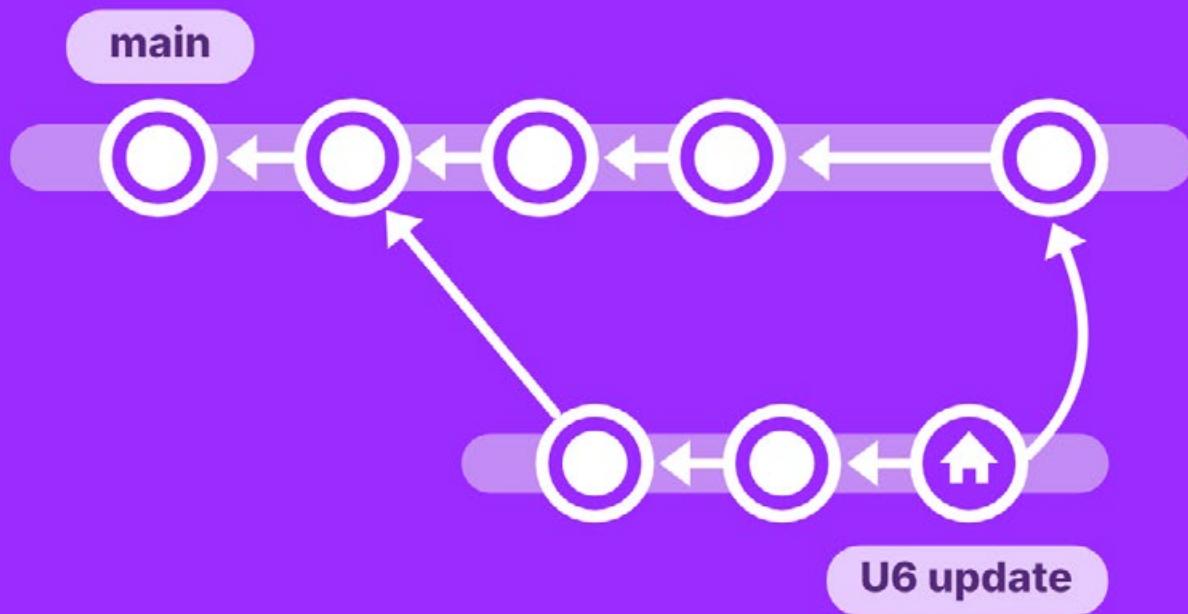


项目组织的最佳实践 和版本控制



介绍

当您从独立工作转变为与团队合作时，软件开发就会变得与众不同。您将项目存储在哪里，以便每个团队成员都可以访问它？如果多人同时处理同一个文件，会发生什么情况？程序员通常了解源代码控制背后的概念，但美术师和其他非技术团队成员呢？您如何最大限度地减少他们需要程序员提供的支持量，以便他们不必担心做错事？

对于游戏开发人员来说，源代码控制或版本控制可能是一个令人生畏的话题，尤其是当您没有技术背景时。但事情并不一定非得是这样的。有许多工具可与 Unity 集成，帮助您的团队有效地进行版本控制。

本指南解释了版本控制的关键概念，比较了一些可用的不同版本控制系统（VCS），并介绍了其他 Unity DevOps 工具，如 Unity Asset Manager、参与和分析、游戏经济、多人游戏服务等。它提供了在设置 Unity 项目时可以使用的提示和技巧，以帮助确保团队协作顺畅高效。最后，您将掌握一些在团队中成功工作的版本控制最佳实践。

源代码控制与版本控制

在计算的开始，所有的软件开发都是纯代码。即使 3D 图形不断发展，一切都仍然被描述为代码。因此，术语“源代码控制”被用来描述用于管理项目内容的系统，而术语源代码管理（SCM）则被作为这些工具的标签。

进入软件和游戏开发的现代时代，我们现在处理的不仅仅是源代码。3D 模型格式（如 FBX、纹理、材质、音频文件等）意味着 SCM 现在必须处理的不仅仅是文本文件更改。术语“源代码控制”不再涵盖我们需要的内容，因此“版本控制系统”或 VCS 成为一个更贴切的描述，现在是所用工具的通用标签。

这两个术语仍然可以互换使用。但是，当谈论经常处理大型二进制资源的 Unity 项目时，版本控制和 VCS 是最准确的，因此在本指南的其余部分将这样引用它们。

最适合 Unity 的三个主要版本控制系统是 Unity 版本控制（UVCS）（以前称为 Plastic SCM）¹、Git 和 Perforce Helix Core。本指南介绍了在团队协作处理 Unity 项目时这些系统的优点和缺点。

¹ Plastic SCM joined the Unity family in 2020, which means that these tools are closely integrated into the Unity Editor.

基本概念

本节介绍版本控制的一些核心概念。如果你不知道你的“commit”和“push”是什么，本节将帮助你理解版本控制术语。

版本控制的工作原理

版本控制允许您保留整个项目的历史记录。它为您的工作带来组织性，并使团队能够高效迭代。

项目文件存储在称为存储库或“存储库”的共享数据库中。您可以定期将项目备份到存储库，如果出现问题，您可以恢复到项目的早期版本。

使用 VCS，您可以进行多个单独的更改，并将它们作为单个组提交以进行版本控制。此提交位于项目时间轴上的一个点，因此，如果您需要恢复到以前的版本，该提交中的所有内容都会被撤消并恢复到当时的状态。您可以查看和修改提交中分组的每项更改，也可以完全撤消提交。

通过访问项目的整个历史记录，可以更轻松地识别哪些更改引入了错误，恢复之前删除的功能，并轻松记录游戏或产品版本之间的更改。

此外，由于版本控制通常存储在云中或分布式服务器上，因此它支持开发团队在任何地方进行协作 - 随着远程工作变得司空见惯，这是一个越来越重要的好处。



为什么要使用版本控制？

除了上述原因之外，版本控制对于进行实验性更改也很有用。您可以在项目的本地版本中添加新功能，如果事情不顺利，您只需恢复您的更改以返回使用干净、功能齐全的项目版本。

你可以迭代实验性的想法，如果你需要帮助解决主项目中的重大问题，版本控制允许你保存更改以备后用。然后，你可以将本地版本返回主分支，以帮助处理任何需要处理的事情。完成后，你可以恢复并继续实验工作。

大多数版本控制系统可以防止您意外覆盖团队中其他人已完成的工作。当您将工作提交到存储库时，您还需要从存储库中“拉取”最新更新。这允许您检查其他人是否没有与您处理同一个文件。这也被称为可怕的“合并冲突”，对于不习惯版本控制的人来说，这可能是可怕的事情之一。但是，一旦您了解了这些工具，通常可以轻松解决合并冲突。Unity 版本控制（UVCS）使用智能锁，通过检查以确保任何分支上的锁定文件是该文件的最新版本，从而帮助降低合并冲突的风险。

集中式与分布式版本控制

在大多数情况下，版本控制系统分为两类之一：集中式或分布式。根据您使用的版本控制系统类型，下面概述的一些术语将适用，一些不适用，有些甚至可能具有不同的含义。让我们来看看这两个类别之间的区别。

集中

集中式系统和分布式系统之间的第一个关键区别是存储库所在的位置。许多公司选择集中式选项，以将托管其专有软件的服务器保留在现场。源代码控制安全性通常是选择此类系统的重要因素。集中式系统不一定意味着现场服务器，因为存储库仍然可以托管在云中，但这种设置比分布式系统少见。





这两种方法之间的另一个主要区别是 *how* 用户将其更改部署到存储库。集中式版本控制通常被视为更直接的选项。使用集中式存储库时，将从存储库获取更改并将其直接发送到存储库。此过程称为 updating from and committing to the repo。

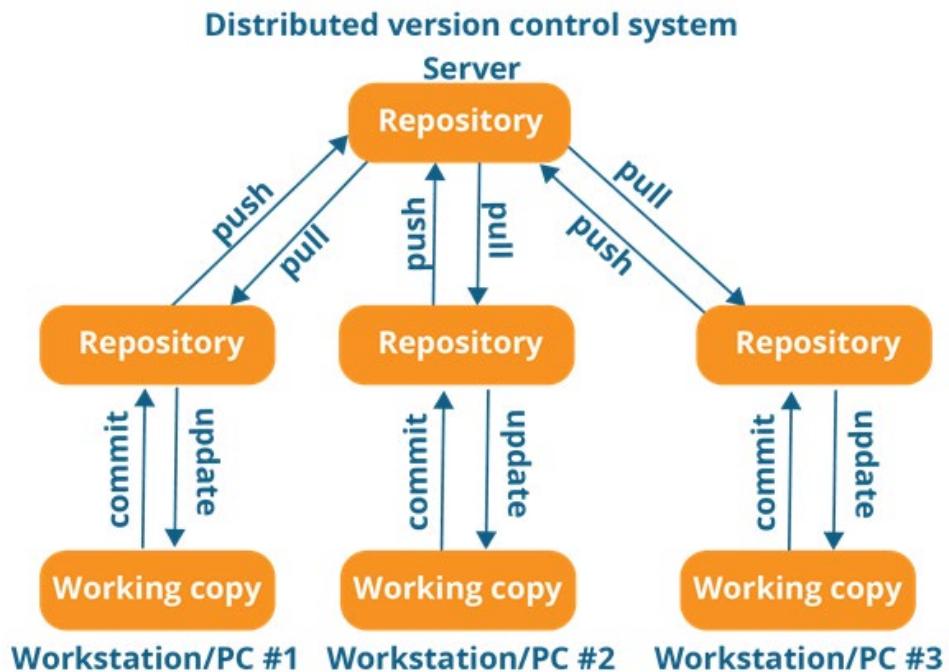
此设置的缺点是用户必须连接到服务器才能提交任何工作。为避免冲突，用户可以锁定文件以进行修改。这称为签出文件，它可以防止其他人提交更改，直到文件被签回。

在集中式工作流程中，用户的工作站上只有最新版本的项目文件，并且服务器保存项目的整个历史记录。

分散式

在分布式工作流中，存储库仍然位于单个位置，通常在 GitHub 等云服务上，但用户将整个项目历史记录克隆到他们的工作站。这允许用户快速处理自己的本地复制和提交更改，因为他们不需要连接到中央服务器。要发送这些更改以便其他人以后可以访问它们，用户需要将它们推送到服务器并拉取任何其他更改。但是，他们不需要像在集中式系统上那样始终使用最新文件。

以这种方式工作允许你创建一组变更集，这些变更集可能等同于一个更大的功能，然后再将它们推给团队的其他成员。事实上，我们鼓励少承诺和经常承诺，但我们稍后会介绍这些最佳实践。





文件锁定在某些分布式工作流程中仍然可用，但是，它不太常见，因为您可以更轻松地处理合并。通过将最新更改从服务器拉取到本地项目，您可以将其他人的更改与您自己的更改进行比较，以确保在将更改推送到存储库之前没有冲突。

虽然分布式方法通常是首选，但它也有一些缺点。首先，在本地计算机上拥有整个项目历史记录会占用大量空间，尤其是对于处理二进制文件类型的团队。Git 有一个称为大文件存储（LFS）的选项，它可以将某些文件的历史记录转换为文本指针，从而减轻一些权重。但是，其他文件具有完整的历史记录，并且存储库最终可能会加载旧的或过时的测试数据。使用小型 M2 驱动器的工作室可能会发现存储库的大小会因旧版本而膨胀，从而使驱动器过载。

其次，由于开发人员不必与中央服务器保持联系，因此他们最终可以长时间孤立工作。他们的本地版本可能会与主存储库完全分离，当需要将他们的更改合并回来时，这可能比他们预期的要多。

典型工作流程

简而言之，以下是集中式系统和分布式系统的工作流程中的步骤：

集中

1. 使用服务器中的更改更新您的工作副本。
2. 进行更改。
3. 将更改提交到中央服务器。

分散式

1. 将任何远程更改拉取到本地存储库中。
2. 进行更改。
3. 提交更改。
4. 根据需要多次执行步骤 2 和 3.
5. 将所有提交推送回远程存储库

本指南重点介绍三个主要的版本控制系统：UVCS、Git 和 Perforce Helix Core。每个 VCS 的详细演练从“版本控制系统”部分开始，但以下是每个 VCS 及其支持的工作流程的简要介绍：

— Unity 版本控制或 UVCS（两者兼而有之）：UVCS 是一个灵活的版本控制系统，具有独特的界面，可为程序员和艺术家提供支持。它擅长处理大型存储库和二进制文件，并且作为基于文件和基于变更集的解决方案，它使您能够仅下载您正在处理的特定文件，而不是整个项目构建。

— Git（分布式）：这是最流行的版本控制系统之一。Git 是开源、免费且灵活的。作为一个平台，Git 是一个仅限命令行的工具。但是已经为它开发了许多不同的 GUI，使用户更容易访问该系统。



Git 和 GitHub 之间经常会有一些混淆。GitHub 是 Git 存储库的托管服务，但您可以在不使用 GitHub 的情况下使用 Git。也就是说，正如任何阅读本文的丰富的开发人员都知道的那样，GitHub 是一项非常受欢迎的服务，因为有一个免费版本（有一些限制），并且它不需要任何自定义服务器设置。

— Perforce Helix Core（集中式）：这是大型游戏工作室通常使用的企业级版本控制系统，因为它具有集中式存储库，这些存储库通常托管在他们自己的服务器上。

关键术语

以下是 VCS 中一些关键功能和流程的术语：

Term	Explanation
Repository/repo	This is the database of all the changes and edits to your project. Stored on a server, either on-site or in the cloud, it holds the full history of the project.
Working copy	This is your local version of the project. Sometimes also called a checkout or workspace. You make changes to your working copy, and, when you're happy with them, commit them to the repository.
Commit/check in	A commit encodes file modifications. A centralized workflow sends those changes to the server and is more commonly called checking in. In a distributed workflow, it adds them to the changeset that needs to later be pushed to the server.
Pull/update/check out	Pulling or updating retrieves the latest changes available on the server. Check out is the more common term when working in a centralized workflow.
Locking	Locking a file prevents it from being edited by another user. You are telling the server, “I’m working on this; please don’t make any other changes.” Locking is generally not supported in distributed workflows.
Clone	In a distributed workflow, cloning a repo is how you initially get a copy of the project and its entire history onto your local machine.
Tags	Tags are special notes that can be added to a commit. They are often used to mark a point in time where a build was made.
Branch	A branch creates a new copy of the codeline, which can then be worked on in parallel. This allows someone to work on parts of the project in isolation, for example a new feature, without affecting the main line of development.



Merge	Merging can happen either when a branch is finished and needs to be merged back into the main line, or even just when two people make changes around the same time. The two changesets will need to be compared and merged together to create the new working copy. Most merges can be handled automatically.
Conflict	A conflict is what happens when merges cannot be handled automatically. This usually occurs when two people have made changes to the same lines of code or the same binary file. Code conflicts can usually be resolved by comparing the text and working out which changes should be accepted, or even whether both can be brought together in a way. For binary files, such as Unity scenes or Prefabs, merging a conflict becomes a lot trickier. However, sometimes a quick conversation with the other contributor is the easiest way to resolve what changes make the most sense to keep.
Pull request	When work on a branch is complete, it's good practice to open a pull request. This signals to the rest of your team that work on that branch is complete and ready to be merged back into the main line. This system gives team leads and/or seniors a chance to review the changes before accepting them back into the main branch.
Head	Head refers to the latest commit on your working copy.
Reset/revert	Depending on your VCS, reset or revert can be used to discard all your local changes back to their state at the head.
Index	The Git index is a file that describes all the current changes you have in your working copy. These changes sit in what's known as the staging area, where you can select which changes you want to add to your next commit.
Git stash	If you have some changes that aren't ready yet for a commit, but you need to move onto some different work, you can use a stash to save those changes in a temporary file and reset your working copy back to head.

组织 Unity 项目的最佳实践

无论您选择哪种 VCS，在 Unity 中工作时，都有一些通常推荐的做法有助于简化版本控制工作流程。首先，让我们看一下您的团队可以有效协作的一些不同方式。

项目组织

文件夹结构

尽管没有单一的方法来组织您的项目，但通常请遵循这些建议。

- 以团队的形式定义并记录您的命名约定和文件夹结构。样式指南和/或项目模板使您的文件更易于查找和组织。选择适合您团队的方法，并确保每个人都参与其中。
- 与您的命名约定保持一致。不要偏离您选择的风格指南或模板。如果您确实需要修改命名规则，请一次性解析和重命名受影响的资源。如果更改影响大量文件，请考虑使用脚本自动执行更新。
- 不要在文件和文件夹名称中使用空格。使用 CamelCase 作为空格的替代项。
- 单独的测试区域或沙盒区域。为非生产场景和实验创建一个单独的文件夹。带有用户名的子文件夹可以按团队成员划分您的工作区域。
- 避免在根级别使用额外的文件夹。通常，将内容文件存储在 Assets 文件夹中。不要在项目的根级别创建其他文件夹，除非它是 assets。



— 将您的内部资产与第三方资产分开。如果您使用的是 Asset Store 或其他插件中的资源，则它们很可能具有自己的项目结构。将您的资产分开。

如果您发现自己正在修改项目的第三方资源或插件，那么当您需要获取插件的最新更新时，版本控制可以真正为您提供帮助。导入更新后，您可以浏览差异以查找您的修改可能被覆盖的位置并重新实现它们。

虽然没有设置文件夹结构，但以下是如何设置 Unity 项目的几个示例。这些结构基于按资产类型拆分项目。资产类型手册页更详细地介绍了最常见的资产。您可以使用 Template 或 Learn 项目作为组织文件夹结构的示例。虽然您不仅限于这些文件夹名称，但它们应该为您提供一个很好的起点。

Example 1

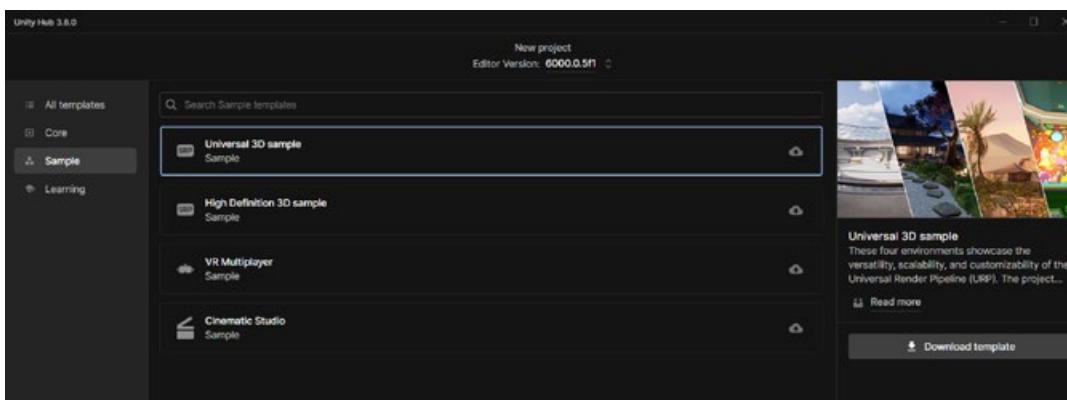
```
Assets
+---Art
| +---Materials
| +---Models
| +---Textures
+---Audio
| +---Music
| \---Sound
+---Code
| +---Scripts  # C# scripts
| \---Shaders  # Shader files and shader graphs
+---Docs    # Wiki, concept art, marketing material
+---Level   # Anything related to game design in Unity
| +---Prefabs
| +---Scenes
| \---UI
```



Example 2

```
Assets
+---Art
| +---Materials
| +---Models
| +---Music
| +---Prefabs
| +---Sound
| +---Textures
| +---UI
+---Levels
+---Src
| +---Framework
| \---Shaders
```

如果从 Unity Hub 下载其中一个模板或初学者项目，您会发现这些项目的子文件夹根据资源类型进行了拆分，如下图所示。



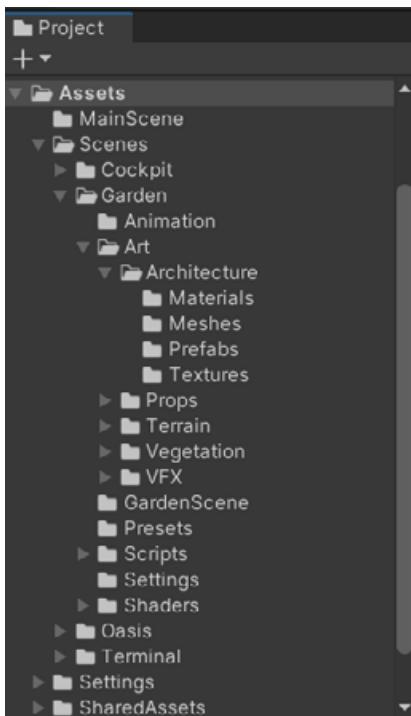
可在 Unity Hub 中下载的模板

根据您选择的模板，您应该会看到代表几个常见资源的子文件夹。以下是按类型组织的一种方法：

Animations	Animations contain animated motion clips and their controller files. These can also contain Timeline assets for in-game cinematics or rigging information for procedural animation.
Audio	Sound assets include audio clips as well as the mixers used for blending the effects and music.
Editor	Here you'll find the scripted tools made for use with the Unity Editor but not appearing in a target build.
Fonts	This folder contains the fonts used in the game.
Materials	These assets describe surface shading properties.



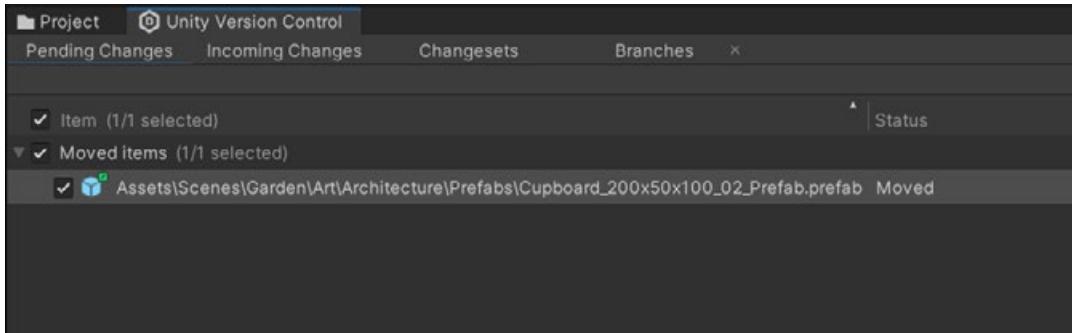
Meshes	Store models created in an external digital content creation (DCC) application here.
Particles	The particle simulations in Unity, created either with the Built-In Particle System or Visual Effect Graph.
Prefabs	These are reusable GameObjects with prebuilt Components.
Scripts	All user-developed code for gameplay appears here.
Scenes	Unity stores small, functional portions of your project into Scene assets. They often correspond to game levels or part of a level.
Settings	This can be used for storing render pipeline settings, such as for the High Definition Render Pipeline (HDRP) and Universal Render Pipeline (URP).
Shaders	These programs run on the GPU as part of the graphics pipeline.
Textures	Image files can consist of texture files for materials and surfacing, UI overlay elements for user interface, and lightmaps to store lighting information.
ThirdParty	If you have assets from an external source like the Asset Store, keep them separated from the rest of your project here. This makes updating your third-party assets and scripts easier. Third-party assets may have a set structure that cannot be altered.
UI	If you're using UI Toolkit, your UXML and USS files are stored here.



带有 URP 模板的示例场景包括许多资源文件夹。

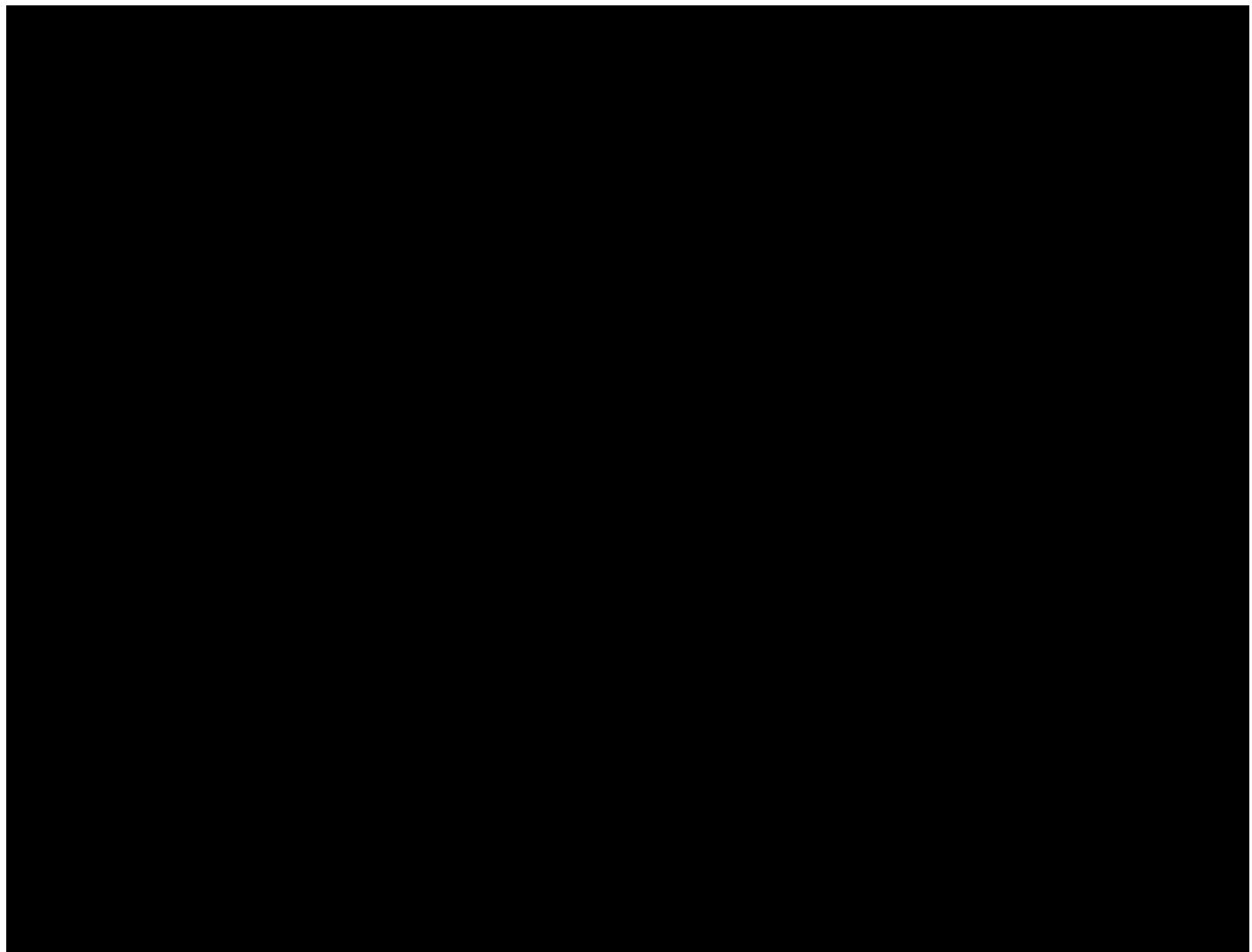
在开始时定义一个好的项目结构将避免以后的版本控制问题。如果您将资产从一个文件夹移动到另一个文件夹，许多 VCS 会将其视为只是删除一个文件并添加另一个文件，而不是正在移动的文件。这将丢失原始文件的历史记录。

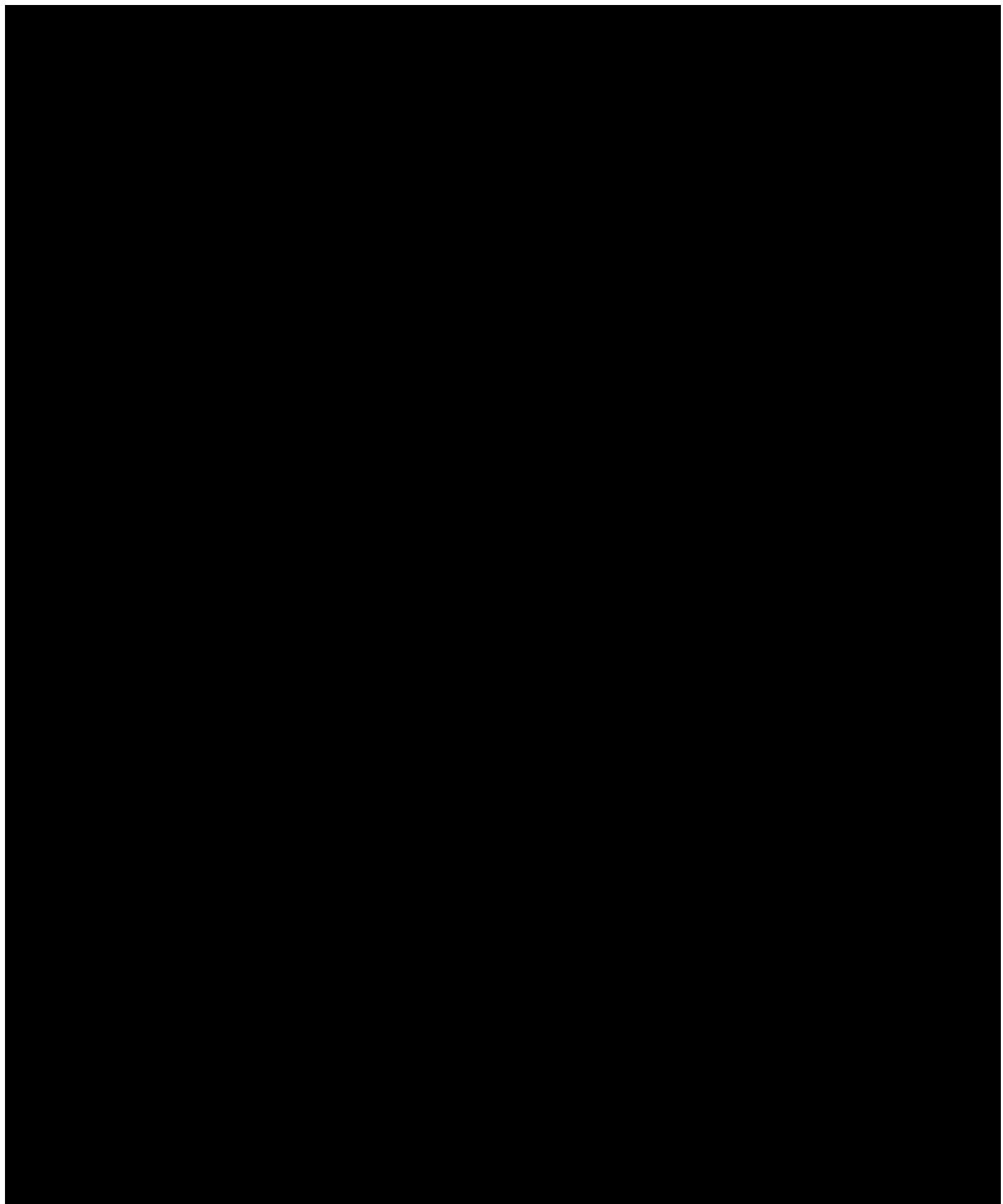
UVCS 可以在 Unity 中处理文件移动，并维护任何移动文件的历史记录。但是，在移动文件时，必须在 Unity Editor 中执行此操作，以便 .meta 文件与资源文件一起移动。

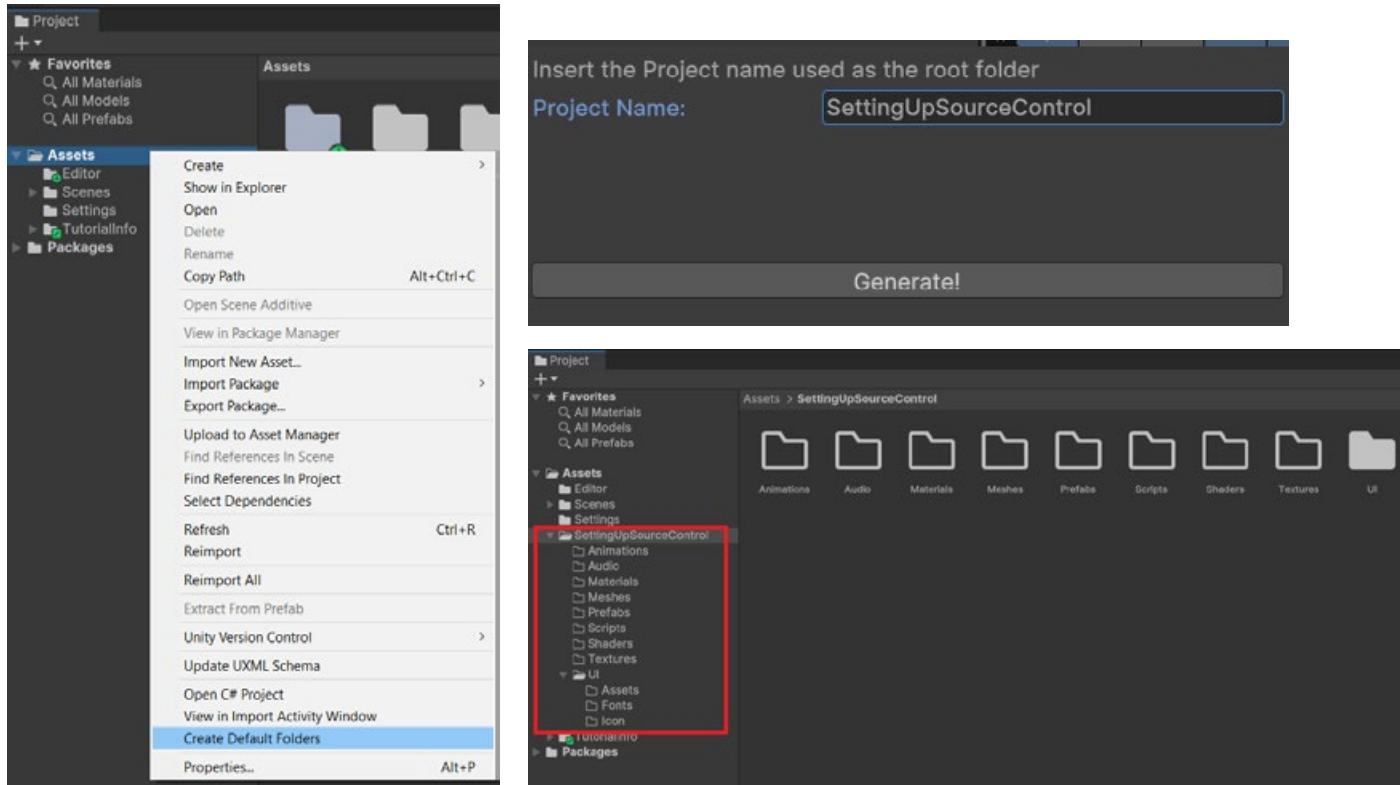


跟踪文件移动

确定项目的文件夹结构后，请使用 Editor 脚本重复使用模板，并为以后的所有项目创建相同的文件夹结构。当它被放置在 Editor 文件夹中时，下面的脚本将在 Assets 中创建一个与“PROJECT_NAME”变量匹配的根文件夹。这样做可以将您自己的工作与第三方软件包分开。







转到菜单 > 资产 > 创建默认文件夹。

在项目开始时创建空文件夹将有助于保持您的团队合作井井有条和高效。

空文件夹

如上图所示的空文件夹可能会在版本控制中带来一些问题 - 因此请仅为您需要的内容创建文件夹。使用 Git 和 Perforce 时，默认情况下会忽略空文件夹。如果这些项目文件夹已设置，并且有人尝试提交它们，则在将某些内容放入该文件夹之前，他们将无法提交。

注意：一种常见的解决方法是将“.keep”文件放在空文件夹中。这足以将文件夹提交到存储库。

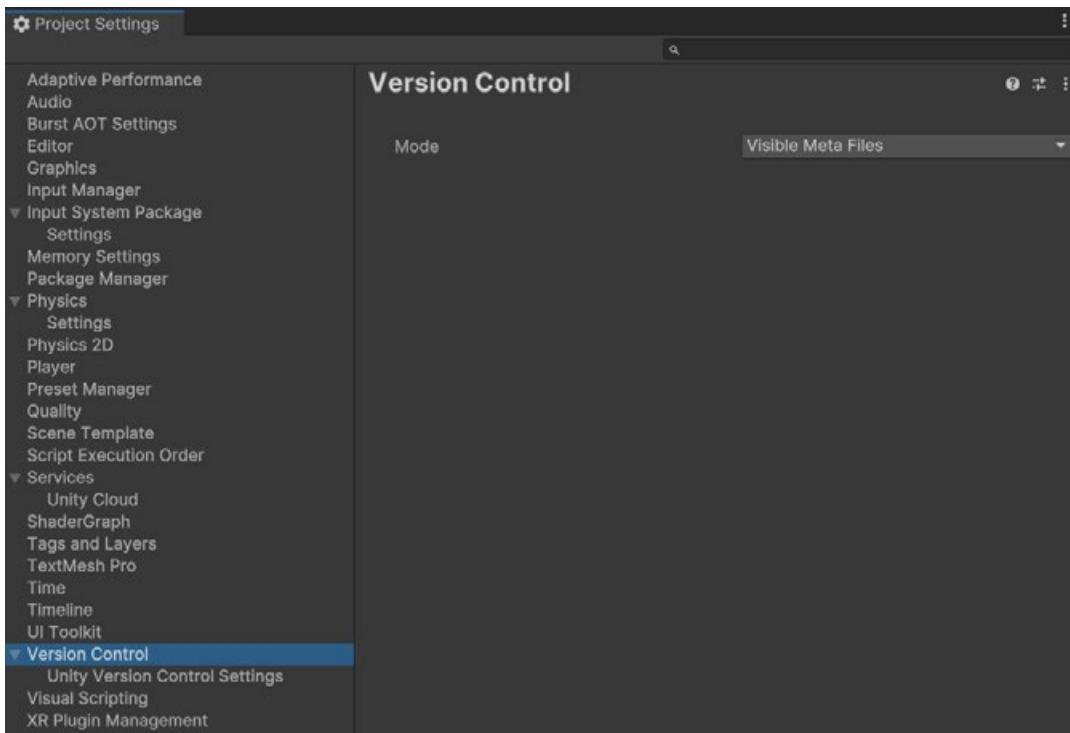
UVCS 可以处理空文件夹。目录被视为实体，并具有与之关联的版本历史记录。

这是在 Unity 中工作时需要注意的一点。Unity 为项目中的每个文件（包括文件夹）生成一个.meta 文件。使用 Git 和 Perforce，用户可以轻松地为空文件夹提交.meta 文件，但文件夹本身最终不会受到版本控制。当其他用户获得最新更改时，其计算机上不存在的文件夹将有一个.meta 文件，然后 Unity 将删除该.meta 文件。UVCS 通过在版本控制下包含空文件夹来避免此问题。



.meta 文件

Unity 会为项目中的每个其他文件生成一个 .meta 文件，虽然通常不建议在版本控制中包含自动生成的文件，但 .meta 文件略有不同。应在 Version Control (版本控制) 窗口中打开 Visible Meta Files (可见元文件) 模式 (除非您使用的是 UVCS 或 Perforce 模式)。



在使用 Git 时打开 Visible Meta Files

虽然 .meta 文件是自动生成的，但它也包含有关与其关联的文件的大量信息。这在具有导入设置的资源（如纹理、网格、音频剪辑等）中很常见。当您更改这些文件的任何导入设置时，更改将写入 .meta 文件，而不是资源文件。这就是您将 .meta 文件提交到存储库的原因，以便每个人都使用相同的文件设置。



```
@@ -3,7 +3,7 @@ guid: 6c9fd0a0956f1984a9785cc7d28a840e
TextureImporter:
internalIDNameTable: []
externalObjects: {}
serializedVersion: 10
serializedVersion: 11
mipmaps:
    mipmapMode: 0
    enableMipMap: 1
@@ -23,6 +23,7 @@ TextureImporter:
isReadable: 0
streamingMipmaps: 0
streamingMipmapsPriority: 0
v1Only: 0
grayScaleToAlpha: 0
generateCubemap: 6
cubemapConvolution: 0
@@ -31,12 +32,12 @@ TextureImporter:
maxTextureSize: 2048
textureSettings:
    serializedVersion: 2
filterMode: #1
aniso: #1
mipBias: -100
wrapU: #1
wrapV: #1
wrapW: #1
filterMode: 1
aniso: 1
mipBias: 0
wrapU: 1
wrapV: 1
wrapW: 1
nPoisScale: 1
lightmap: 0
compressionQuality: 50
@@ -49,18 +50,22 @@ TextureImporter:
spriteBorder: {x: 0, y: 0, z: 0, w: 0}
spriteGenerateFallbackPhysicsShape: 1
alphaUsage: 1
alphaIsTransparency: 0
alphaIsTransparency: 1
spriteTessellationDetail: -1
textureType: 0
```

调整文件的导入设置时对 .meta 文件的更改

命名标准

就标准达成一致并不止于项目文件夹结构。为场景中的游戏对象或工程文件夹中的预制件设置命名标准，可以使您的团队在最终处理彼此的文件时更容易理解。

虽然游戏对象没有明确的命名标准，但请考虑以下事项。

Standard	Example
<p>Use descriptive names and don't abbreviate.</p> <p>Use names that you will remember several months from now. Consider whether another person will understand your notation, and choose names that you can pronounce and remember. For this reason abbreviations are generally not recommended.</p>	largeButton, LargeButton, or leftButton NOT: IButton
<p>Use Camel case/Pascal case.</p> <p>Avoid spaces in your object names. Camel case or Pascal case improve readability (and typing accuracy according to this study).</p>	OutOfMemoryException, dateTimeFormat, NOT: Outofmemoryexception, datetimeformat



<p>Use underscores (or hyphens) sparingly. Avoid underscores and hyphens in general. However, they can be useful in certain circumstances. Prefixing a name with an underscore puts it alphabetically first. You can also use underscores to denote variants of a specific object.</p>	<p>Active States: EnterButton_Active, EnterButton_Inactive Texture Maps: Foliage_Diffuse, Foliage_Normalmap Level of Detail: Building_LOD1, Building_LOD0</p>
<p>Use number suffixes to denote a sequence. Likewise, don't suffix with a number if it's not part of a list.</p>	<p>For a path, name the nodes: Node0, Node1, Node2, etc.</p>
<p>Follow the design document naming.</p>	<p>If your design document names locations like HighSpellTower or RedDragonLair, use those exact spellings.</p>

在整个团队中就一致的风格达成一致，可以使项目更简洁、更具可读性和可扩展性。Unity 电子书 *Create a C# style guide: Write cleaner code that scales* 提供了有关命名约定、格式、类、方法、注释等的提示和最佳实践。总体而言，该指南遵循 Microsoft C# 样式标准 2，提供了特定于 Unity 的子集，但没有一个“真正的”方法;Google C# 2 指南也是定义有关命名、格式和注释约定的准则的绝佳资源。

工作流程优化

除了在 Assets 文件夹中保存资源的方式和位置之外，您还可以做出多种设计和开发选择来帮助加快工作流程，尤其是在使用版本控制时。

拆分您的资产

大型单个 Unity 场景不适合协作。将关卡分解为许多较小的场景，以便艺术家和设计师可以在单个关卡上更好地协作，同时最大限度地降低冲突风险。

在运行时，您的项目可以使用 SceneManager.LoadSceneAsync 传递 LoadSceneMode.Additive 参数模式以附加方式加载场景。

此外，尽可能将工作分解为预制件。如果以后需要进行更改，可以更改预制件而不是使用它的场景，以避免与在场景中工作的任何人发生冲突。在版本控制下执行差异时，预制件更改通常更容易阅读。

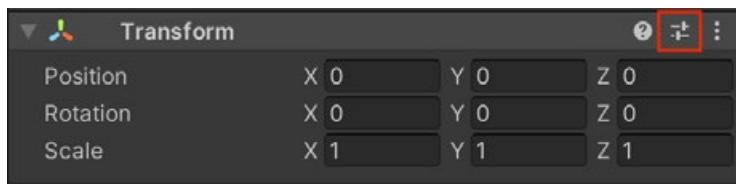


如果最终出现场景冲突，Unity 还有一个内置的 YAML（一种人类可读的数据序列化语言）工具，专门用于合并场景和预制件。有关更多信息，请参阅 Unity 文档中的智能合并。

预设

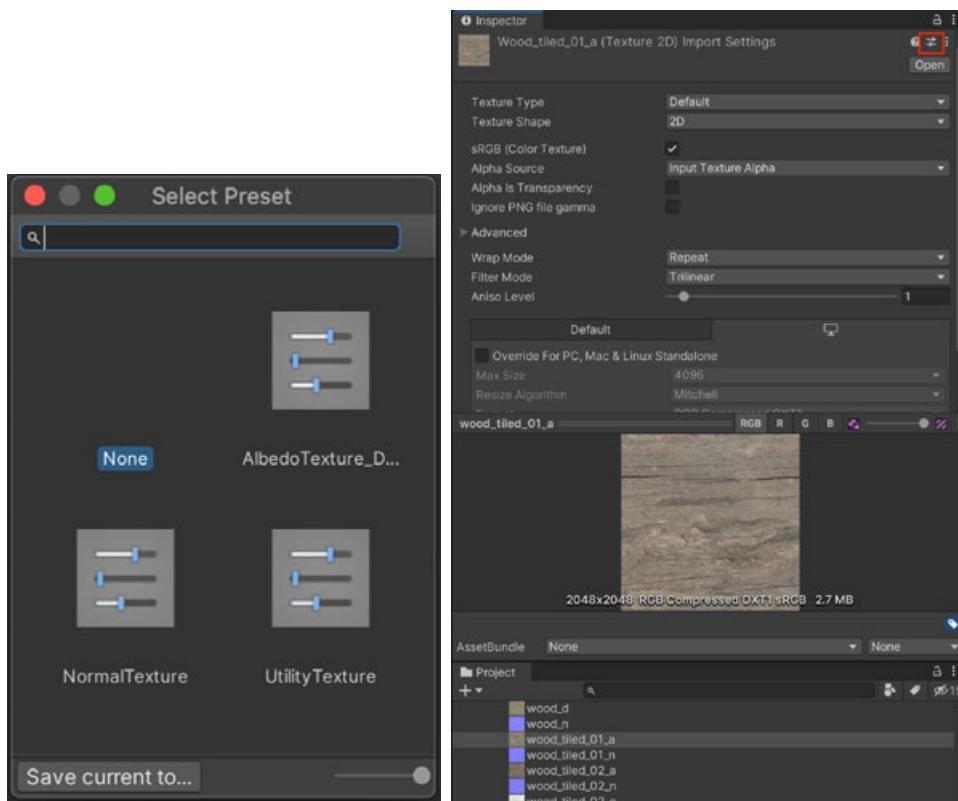
Unity 预设是预定义的配置，允许您在项目中的不同组件、资源或工具中保存和重复使用特定设置。通过创建预设，您可以复制组件或资源的设置，将其保存为资源，然后稍后将相同的设置应用于另一个项目。

使用预设可强制实施标准或将合理的默认值应用于新资源。这可确保整个团队的标准一致，因此通常被忽视的设置不会影响项目的性能。



预设图标在此处以红色高亮显示

单击组件右上角的 Preset 图标。单击 Save current to... 将预设保存为资产。单击其中一个可用的预设以加载一组值。



在此示例中，预设包含 2D 纹理的不同导入设置，具体取决于用途（反照率、法线或实用程序）。



使用预设的其他便捷方法包括：

- 创建具有默认值的游戏对象：将 Preset 资源拖放到 Hierarchy 中，以使用包含 Preset 值的相应组件创建新的游戏对象。
 - 将特定类型与预设关联：在预设管理器（项目设置 > 预设管理器）中，为每个类型指定一个或多个预设。然后，创建新组件将默认为指定的 Preset 值。
 - 专业提示：为每个类型创建多个预设，并依靠过滤器按名称关联正确的预设。
 - 保存和加载管理器设置：对管理器窗口使用预设，以便可以重复使用设置。例如，如果您计划重新应用相同的标签和图层或物理设置，预设可以缩短下一个项目的设置时间。

代码标准

编码标准还将有助于保持团队工作的一致性，并使开发人员更容易在项目的不同领域之间切换。同样，这里没有一成不变的规则。您需要决定什么最适合您的团队——但一旦你决定了，就坚持下去。

例如，命名空间可以帮助更好地组织您的代码。它们允许您在项目中分隔模块，并避免与第三方资源发生冲突，因为类名称最终可能会重复。

在代码中使用命名空间时，请按命名空间拆分文件夹结构，以便更好地组织。

标准标头也是一种很好的做法。在代码模板中包含标准标头将有助于记录类的用途、创建日期，甚至 `who` 创建类。所有这些都很容易在项目的漫长历史中丢失，即使在使用版本控制时也是如此。

Unity 使用模板脚本，每当您在项目中创建新的 Monobehaviour 时，都会从中读取。每次创建新脚本或着色器时，Unity 都会使用存储在 `%EDITOR_PATH%\Data\Resources\ScriptTemplates` 中的模板：

— Windows: `C:\Program Files\Unity\Editor\Data\Resources\ScriptTemplates` — Mac: `/Applications/Hub/Editor/[version]/Unity/Unity.app/Contents/Resources/ScriptTemplates`

默认的 Monobehaviour 模板是这样的：81-C# Script-NewBehaviourScript.cs.txt

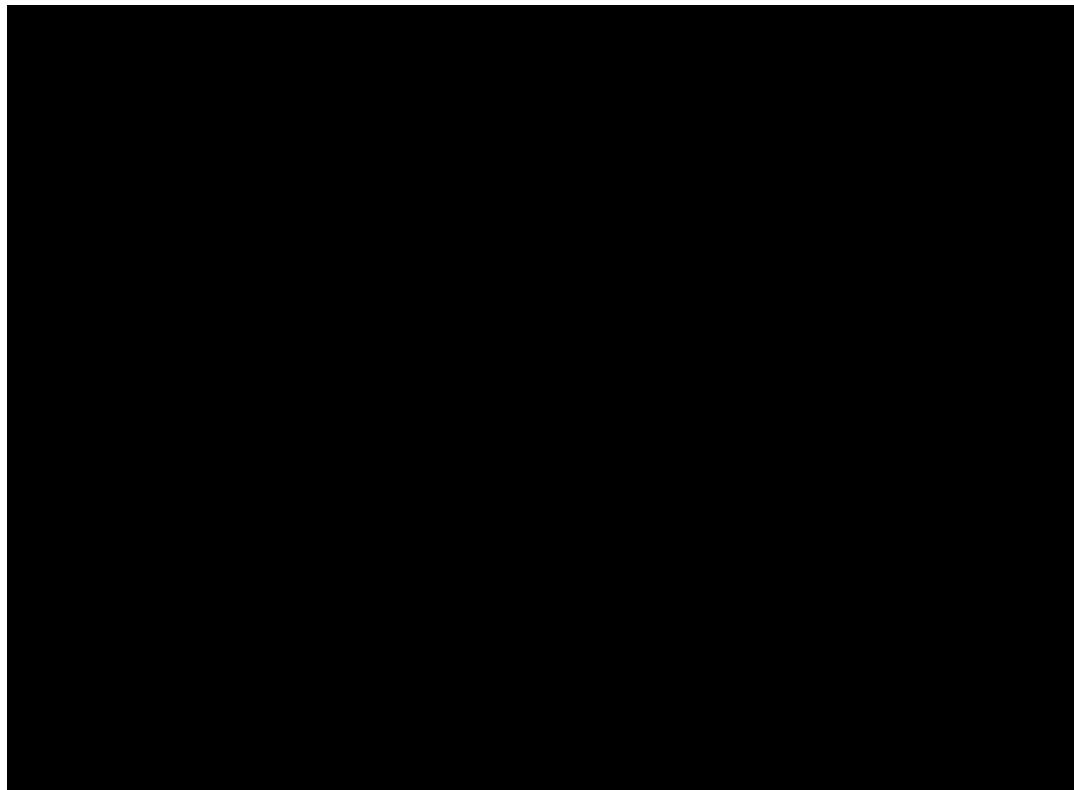


还有用于着色器、其他行为脚本和程序集定义的模板。

对于特定于项目的脚本模板，请创建一个 Assets/ScriptTemplates 文件夹，并将脚本模板复制到此文
件夹中以覆盖默认值。

您还可以直接修改所有项目的默认脚本模板，但请确保在进行任何更改之前备份原始脚本模板。Unit
y 的每个版本都有自己的模板文件夹，因此当您更新到新版本时，需要再次替换模板。

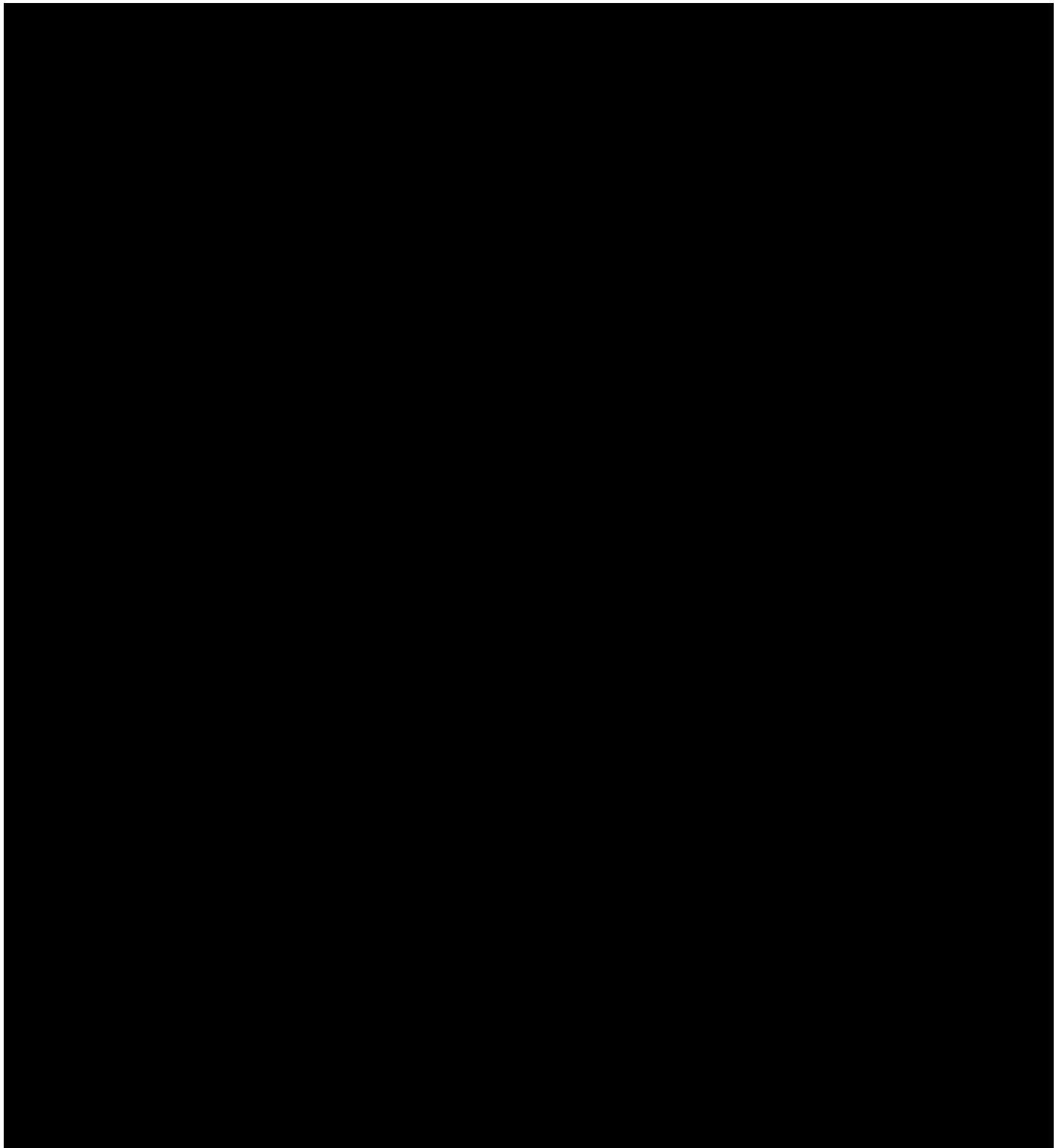
原始的 81-C# Script-NewBehaviourScript.cs.txt 文件如下所示：



有两个关键字可能会有所帮助：

- #SCRIPTNAME# 表示输入的文件名或默认文件名（例如，NewBehaviourScript）。
- #NOTRIM# 确保括号中包含一行空格。

您还可以使用自己的关键字，并将其替换为实现 OnWillCreateAsset 方法的 Editor 脚本。



您还可以使用自己的关键字，并将其替换为实现 OnWillCreateAsset 方法的 Editor 脚本。



UI Toolkit 格式约定

UI Toolkit 使用 UXML 代码。由于受到标准 Web 技术的启发，UI Toolkit 使用 kebab-case（也称为破折号或 lisp-case）来命名类，与 CSS 相关样式系统中使用的约定相同。

就像类一样，HTML（以及扩展的 UXML）通常使用 kebab-case（小写带破折号）作为 ID 名称。因此，像 my-element-id 这样的名称会更标准。

但是，建议 ID 名称具体且具有描述性，以便轻松识别元素的用途。例如，submit-button 或 main-nav-container、logo-image 等。

	Recommended	Not recommended
Classes	.menu-bar-blue	ButtonBlue
Visual Element IDs	main-nav-image	

UXML 中建议的命名约定

使用 kebab 风格的主要原因包括：

- CSS 选择器不区分大小写，因此使用 Camel 大小写或 Pascal 大小写（如 .buttonBlue 或 ButtonBlue）可能会导致混淆。
- HTML 规范建议将属性名称全部小写，即 .button-blue，而不是 .buttonBlue 或 .ButtonBlue。
- 当涉及多个单词时，Kebab 大小写很容易阅读和写入。

无论您选择使用哪种命名约定，在代码库中保持一致都很重要。所有类型的脚本的一致命名约定可以使代码更简洁，并使所有团队成员更容易访问和修改。

项目组织服务

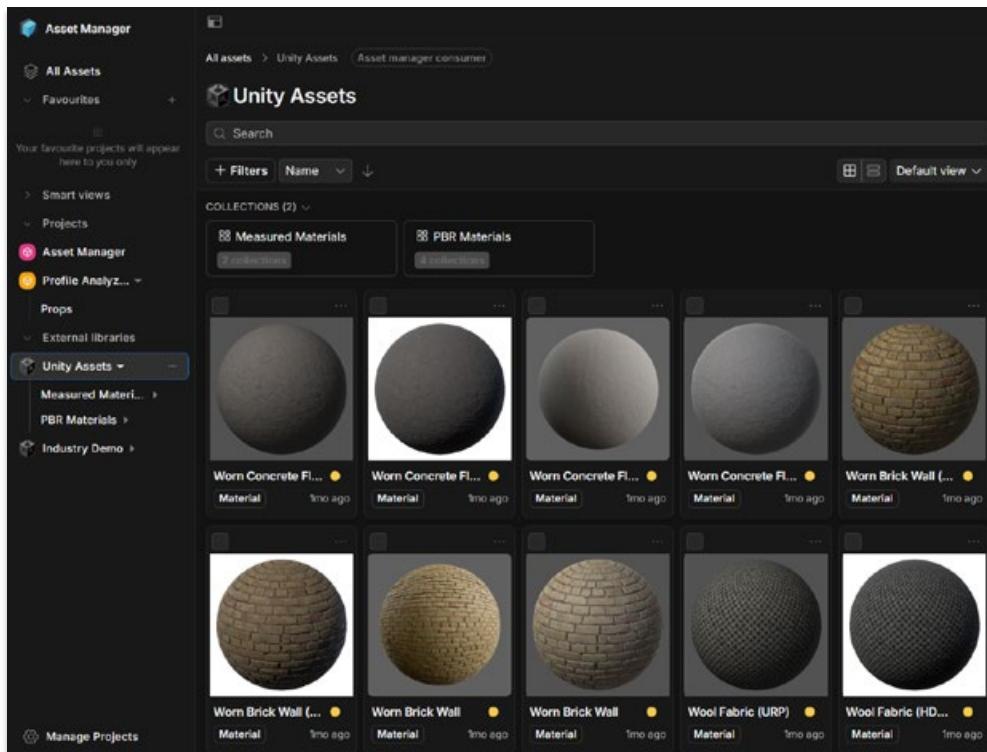
资产管理器

Unity Asset Manager 是 Unity 基于云的可扩展数字资产管理（DAM）解决方案，可让您提高整个组织中内容的可发现性、重用性和投资回报率。Asset Manager 允许您使用云存储为 Unity 项目中的内容编制索引，使整个团队即使位于不同位置也能轻松发现资产。它的主要功能包括：

- 更好的资源管理 — 直观的浏览和发现 — 与创意工具的基本集成
- 生命周期管理 — 基于角色的权限
- 灵活性和可扩展性

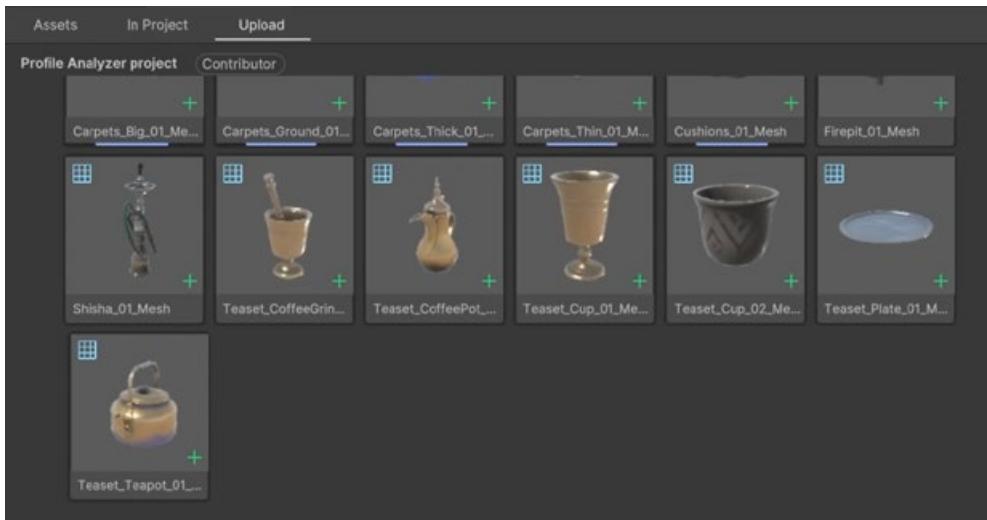


登录 Unity Cloud 时，您可以通过仪表板找到一系列免费的材质、纹理和演示。打开 Unity Assets 子部分，将这些内容下载到您的项目中。



Unity Cloud 上 Unity Assets 部分中的免费材质和纹理

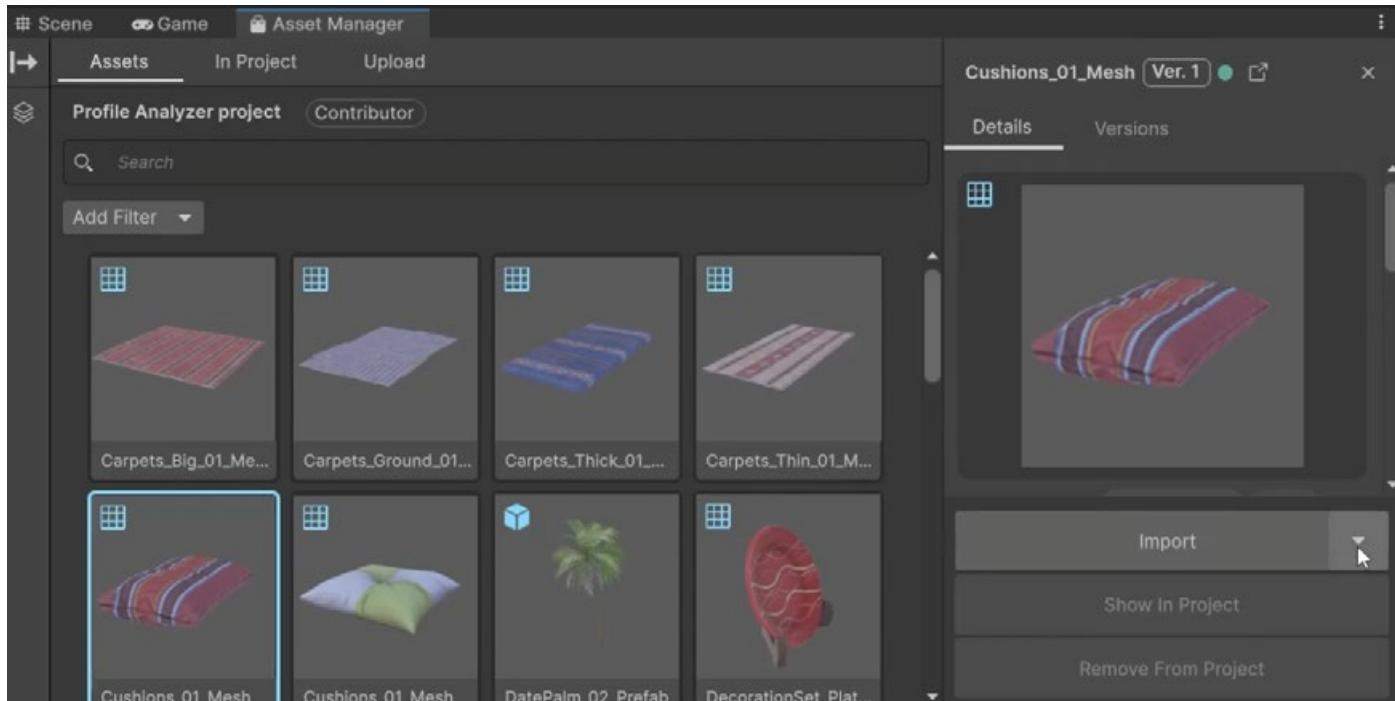
在 Unity 中，您可以将文件从当前项目拖动到 Asset Manager 窗口中，以将其上传到云中。然后，这些文件可供使用 Asset Manager 的任何其他项目以及使用 UVCS 连接到项目的团队成员使用。



将资产拖动到 Upload 选项卡以上传到云。

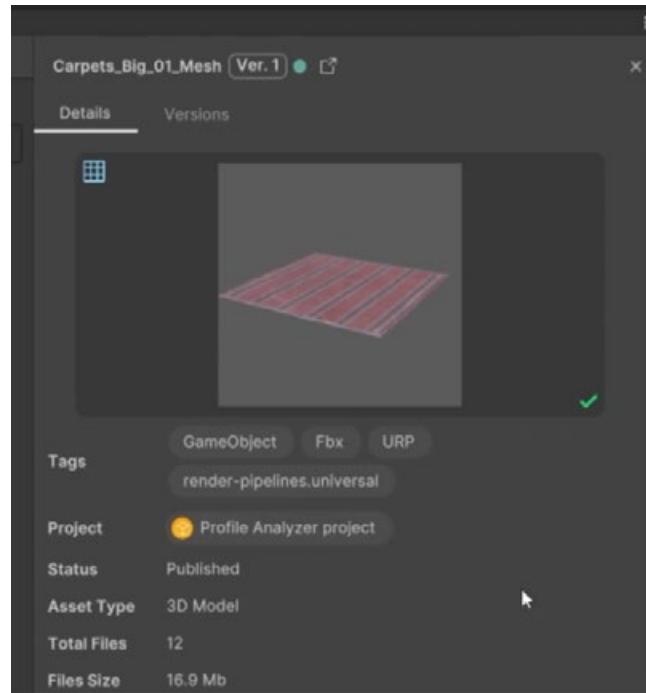


然后，您可以直接从 Asset Manager 窗口将文件导入到其他项目中，使该系统易于使用并全面存档所有项目资源。



单击 import (导入) 以下载文件并将其导入到您的项目中。

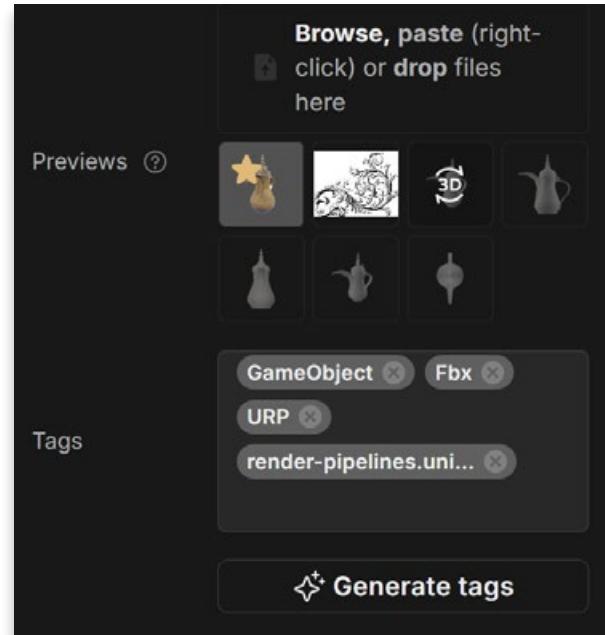
标记是自动生成的，以便在您拥有大量资源时启用快速搜索。资源也链接到项目，因此您可以将搜索限制为属于特定项目或艺术家的资源。



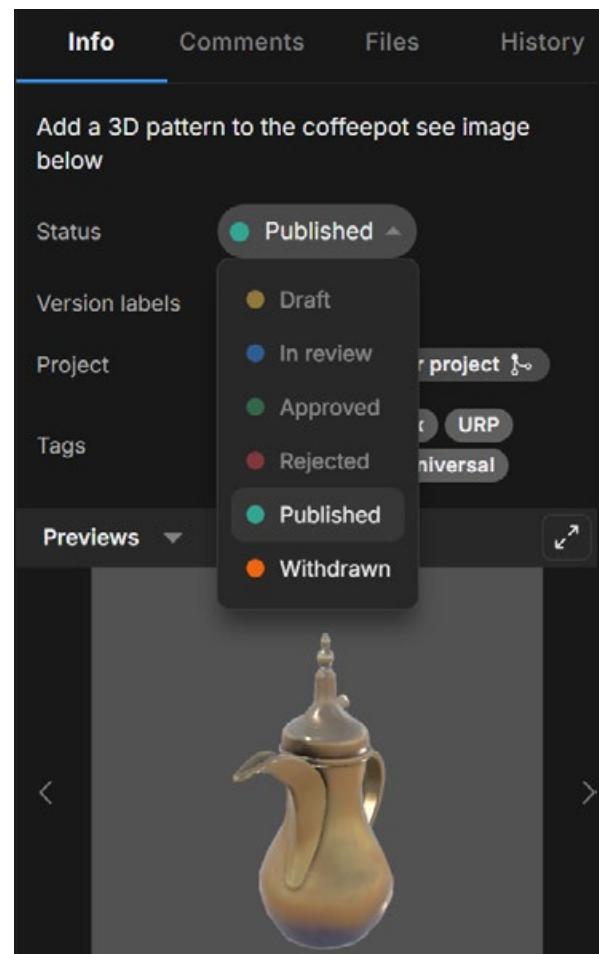
自动生成的标签使搜索资产变得快速而简单。



可以在 Unity Cloud 中编辑资源。添加自定义缩略图、额外的图像预览、更多标签和评论。



编辑资源以添加额外的图像预览和标记。

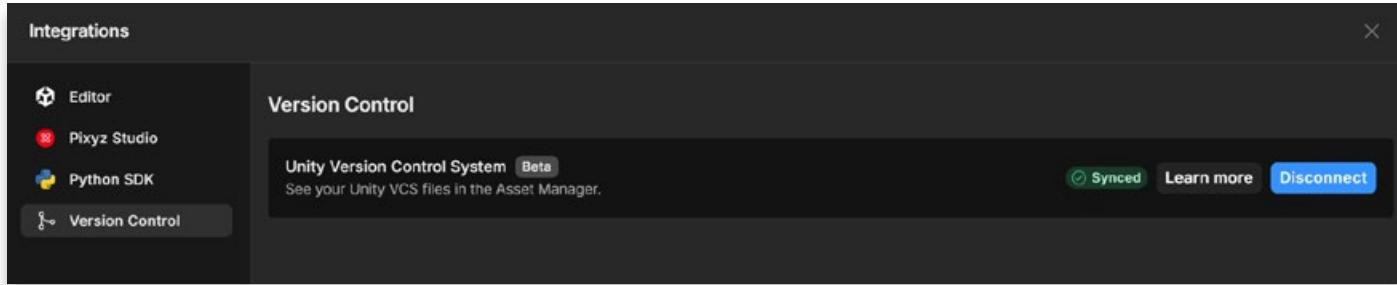


更改资产的状态。



团队成员始终可以访问最新版本，从而减少使用资产迭代时的问题。

Asset Manager 通过在 Cloud 上启用集成来与 UVCS 配合使用



将 Asset Manager 集成到 UVCS 以同步数据。

通过 UVCS 所做的任何修改都将自动与 Unity Asset Manager 同步，从而防止在团队中创建冗余内容。

详细了解 Unity Asset Manager her 的优势

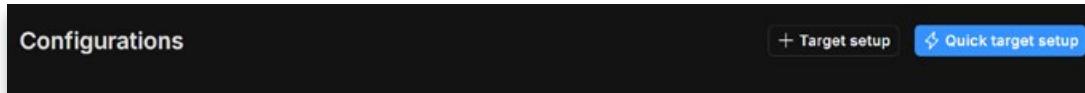
e.

您还可以在 Unity Learn 上观看此 Asset Manager 教程。

构建自动化

构建自动化是任何 DevOps 策略不可或缺的一部分。您可以在云上运行构建，也可以为所有目标平台同时运行构建。您无需等待构建在您的设备上按顺序完成，而是可以将这些时间花在创建项目上。

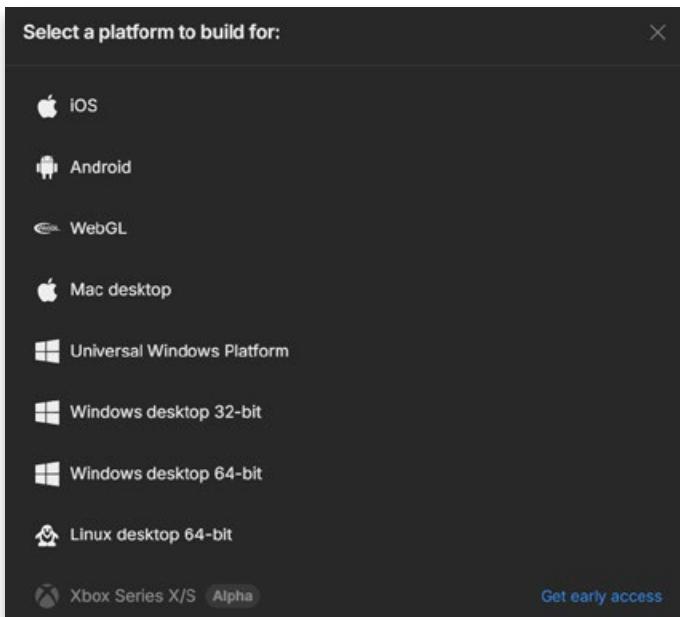
在 Unity Cloud 控制面板中，您可以从 Configurations 部分设置任意数量的构建目标



单击 Quick target setup 以创建 Build 目标（或者，也可以单击 Target setup（如果要配置更高级的选项），也可以单击 Target setup）。

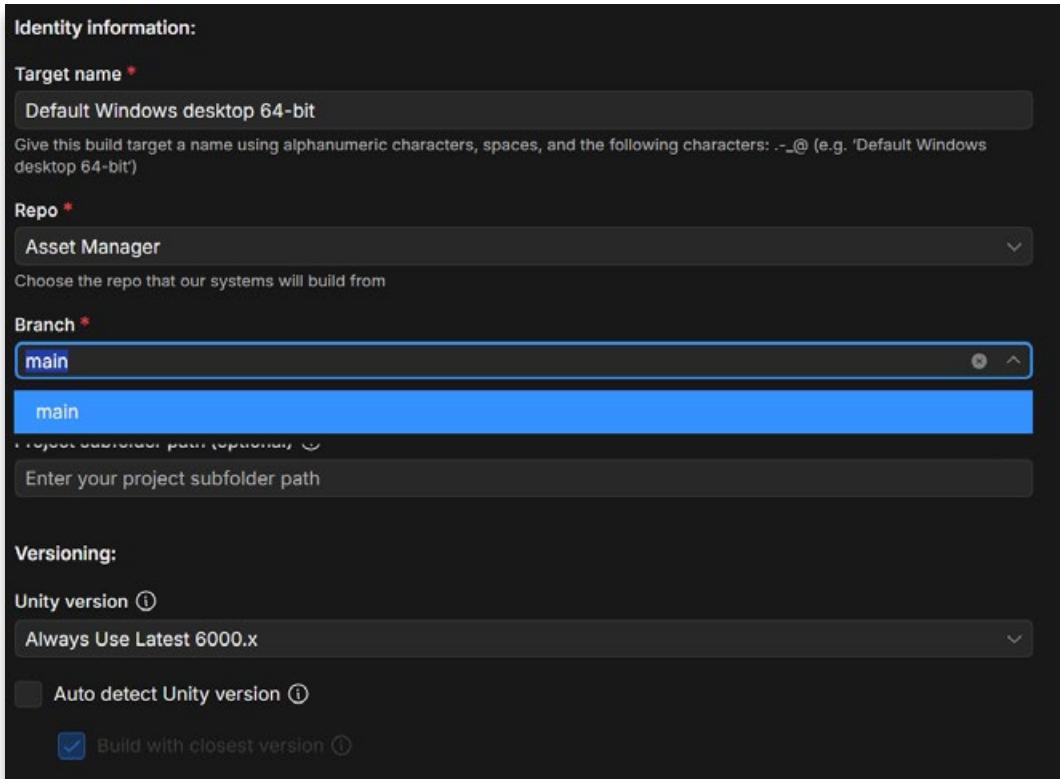


选择要为其构建的平台。您可以为每个平台设置多个构建目标。



要为其构建的目标平台

您可以从任何分支构建，并为每个分支设置多个目标。您还可以选择特定版本的 Unity。

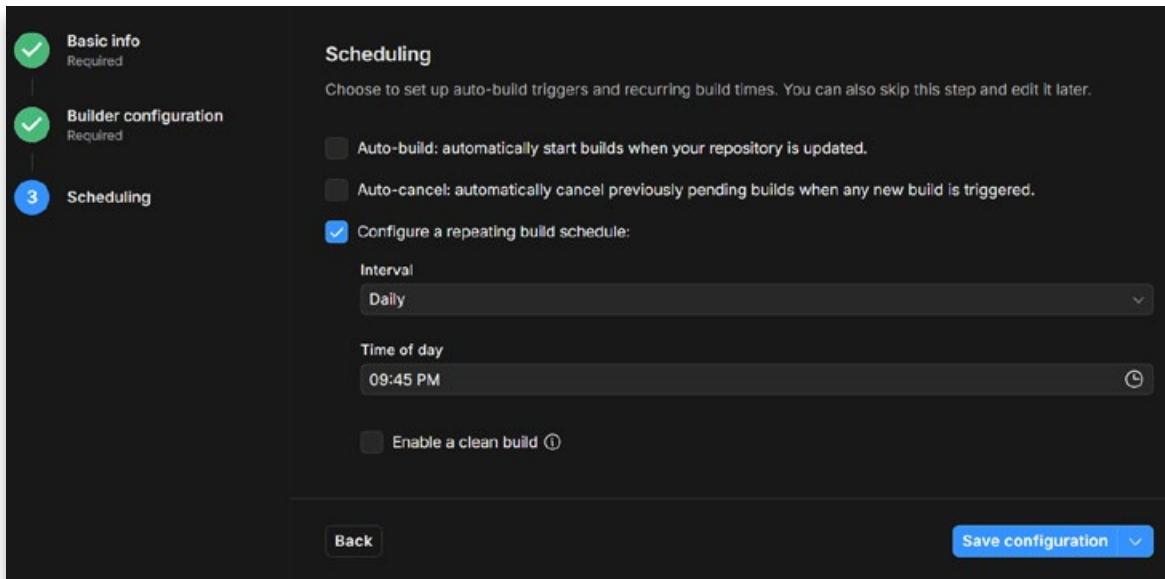


选择要从中创建生成的分支。



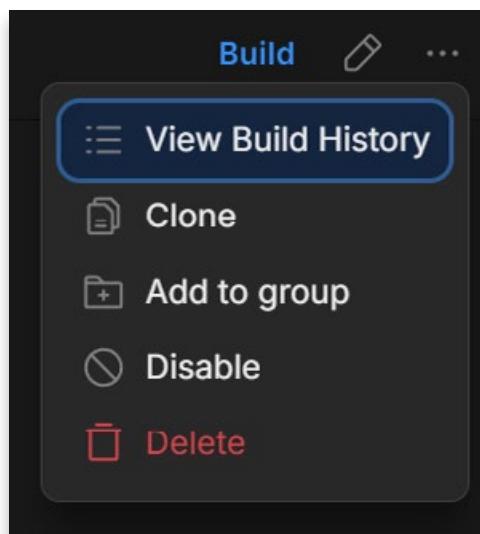
注意：在撰写本文时，Builder 作系统中的 Windows 11 每月提供 200 分钟的免费构建时间。

然后，您可以设置计划。您可以构建一次来测试您的进度，也可以设置一个重复的计划。以下示例显示了一个每日计划，该计划将在晚上 9:45 生成生成版本，以便每天早上提供最新的游戏生成版本以检查是否存在任何问题或错误。构建完成后，您将收到一封电子邮件，以指示其状态，无论是成功还是失败。



为您的构建创建计划

在 configurations（配置）部分中，您可以单击 Build（构建）旁边的设置以立即构建。您还可以暂停或删除构建配置。



禁用或删除构建计划。

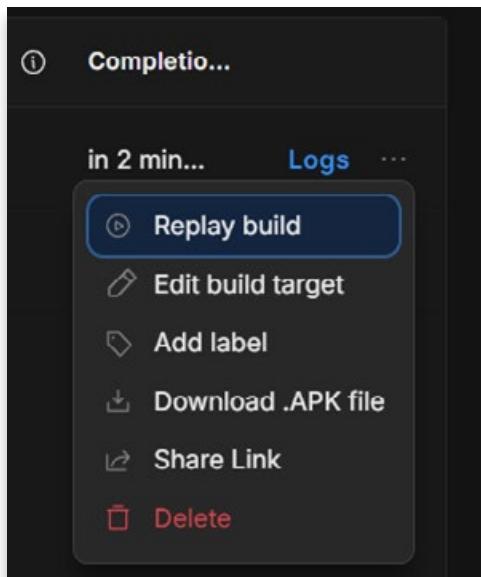


在 Build Automation 控制面板的 Build History（构建历史记录）部分中，您可以检查构建的结果。如果其中任何一个失败，您可以通过检查日志来排查故障。

The screenshot shows the Unity Build History interface. On the left, there's a sidebar with various navigation options like Overview, Explore repositories, Version Control, etc. The Build Automation section is expanded, and Build history is selected. The main area is titled 'Build History' and shows 'View Build History for All Projects'. It displays summary statistics: Build health (2 successful, 0 failed), Average build time (00:36:21 including wait time), Average build size (36.47 MB), and Concurrent builds (1). Below this are dropdown menus for Build Target, Status, Display, Platform, and Build Target Group. A search bar is also present. A table lists two build entries: one for Android (Default And...) and one for Windows (Default Win...). Both builds are marked as successful (#1) and completed. The table columns include Status & #, Build target, Platform, Build time, Completion status, and Log links.

在云中构建历史记录

您可以从云中下载、共享或删除内部版本。



下载、共享或删除内部版本。



如果您希望在构建完成后将消息发送到其他应用程序，例如 Discord 或 Slack，请设置集成。

转到 Administration 然后转到 Project Integrations on the Cloud，然后单击 New Integration。

The screenshot shows the Unity Cloud interface. On the left, a sidebar lists various management options like Dashboard, Projects, Products, Administration, and DevOps. The 'Administration' section is currently selected. Under 'Administration', the 'Project Integrations' option is highlighted. A modal window titled 'New Integration' is open over the main content area. The modal has three steps: 'Select an integration', 'Select an event', and 'Configure options'. Step 1 is active, showing a list of integration types: Webhook, Discord, Slack, Email, JIRA, Trello, and Microsoft Teams. The 'Slack' option is selected. Below the list is a note: 'Send notifications to your team's Slack channel.' At the bottom right of the modal is a 'NEXT' button.

通过管理 > 项目集成添加新的集成。

选择您要使用的应用程序。现在，将通过这些应用程序发送消息以通知团队成员。

Learn more about Unity Build Automation.

版本控制系统

现在，您已经熟悉了版本控制、项目组织和命名约定最佳实践中的一些关键术语和概念，是时候介绍一些关键参与者了。当然，没有一种解决方案适合所有人。在选择在团队中使用哪种 VCS 时，需要考虑很多事情。希望在本书结束时，您将拥有做出该决定所需的所有信息。

Git 公司

Git 开源、免费且灵活，是最受欢迎的版本控制系统之一。但是，作为分布式设置，对于非技术用户来说，它可能会令人生畏。

由 Linus Torvalds 于 2005 年开发以控制 Linux 内核开发，此后一直保持良好维护和开源。Git 作为平台是仅限命令行的工具。但是已经为它开发了许多不同的 GUI，使用户更容易访问该系统。

Git 和 GitHub 之间经常会有一些混淆。GitHub 是 Git 存储库的托管服务，但您可以在不使用 GitHub 的情况下使用 Git。也就是说，GitHub 是一项非常受欢迎的服务，因为有一个免费版本（有一些限制），并且它不需要任何自定义服务器设置。



The screenshot shows the GitHub profile for Unity Technologies. It features a header with the GitHub logo, search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header is the Unity Technologies logo and information: Copenhagen, Denmark, http://unity.com, and a Verified badge. The main content area displays a grid of popular repositories, each with a thumbnail, name, description, and statistics (stars, forks, issues). To the right, there are sections for People (a grid of user avatars), Top languages (C#, C++, Python, Go, JavaScript), and Most used topics (unity, unity3d, robotics-simulation, reinforcement-learning, unity-robotics). A sidebar on the left contains a 'Repositories' section with a search bar and filters for Type, Language, and Sort.

GitHub

一些流行的 Git GUI 客户端包括：

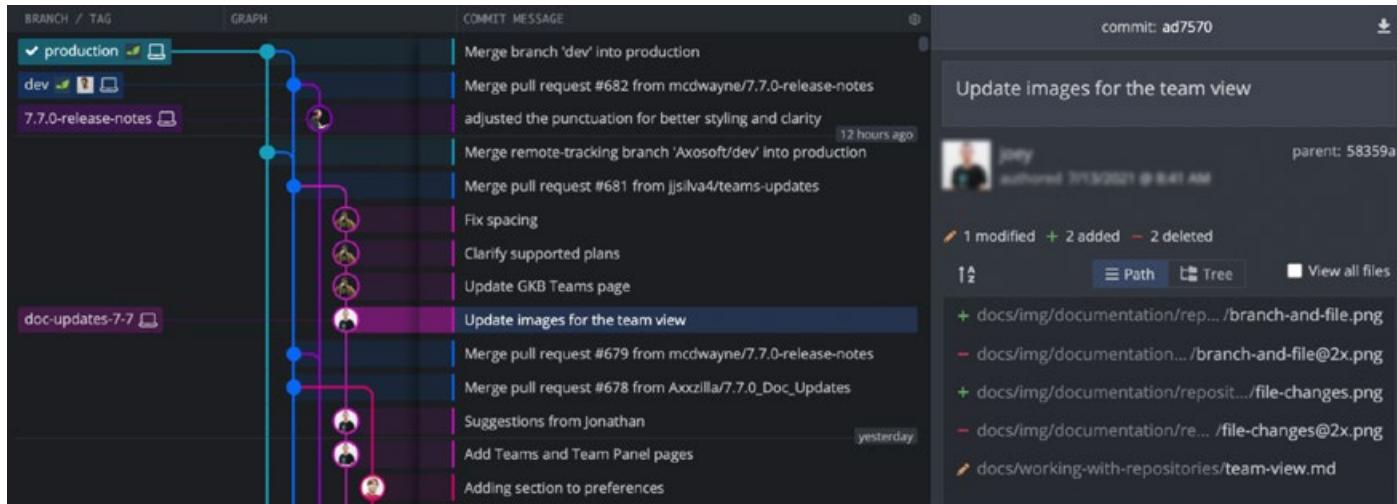
Fork: 这是一个快速友好的 GUI，您可以下载以进行免费评估。

The screenshot shows the Fork Git GUI client interface. The top menu bar includes File, View, Repository, Window, and Help. The main window has tabs for Fork, TypeScript*, master, Bootstrap, and Imgui. The Fork tab shows a commit history for the 'TypeScript*' branch, with the 'master' commit selected. The commit details show a pull request (#43475) adding @link jsdoc auto-complete. The Changes tab shows the diff for this commit. The File Tree tab shows the directory structure of the src/compiler/parser.ts file. On the left, a sidebar lists branches (Starred: master, Branches: aozgaa, dev, lego, rbuckton, master, persistentResolutions, release-4.1), remotes (origin), tags, stashes, and submodules. On the right, a list of commits is shown, each with a committer's name, commit hash, date, and time. At the bottom, there are buttons for Commit, Changes, File Tree, and Expand All.

Fork

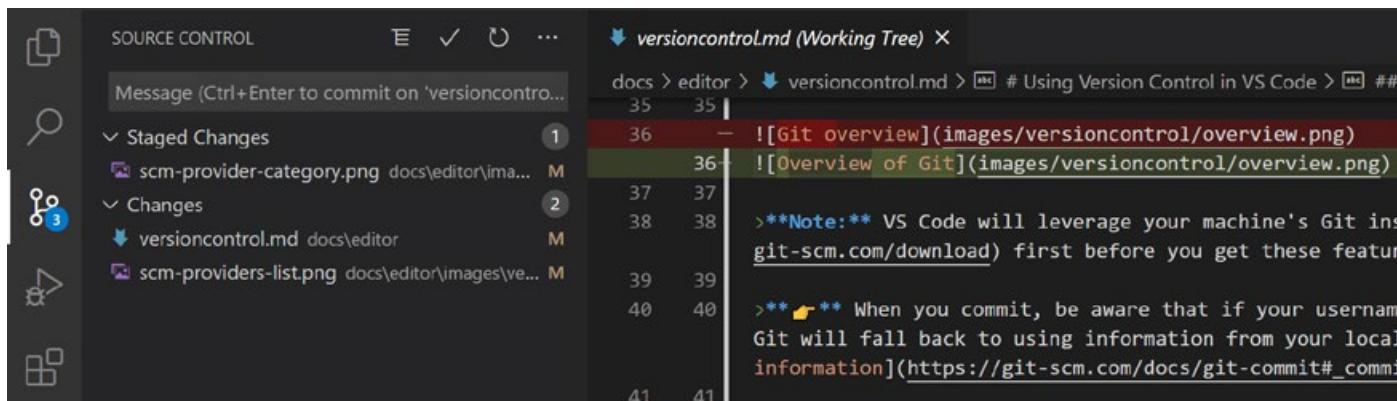


GitKraken：这提供了一种更直观、更易于访问的 Git 使用方式，具有直观的 UI，并且可以灵活地在 GUI 或 CLI 终端之间切换。



GitKraken

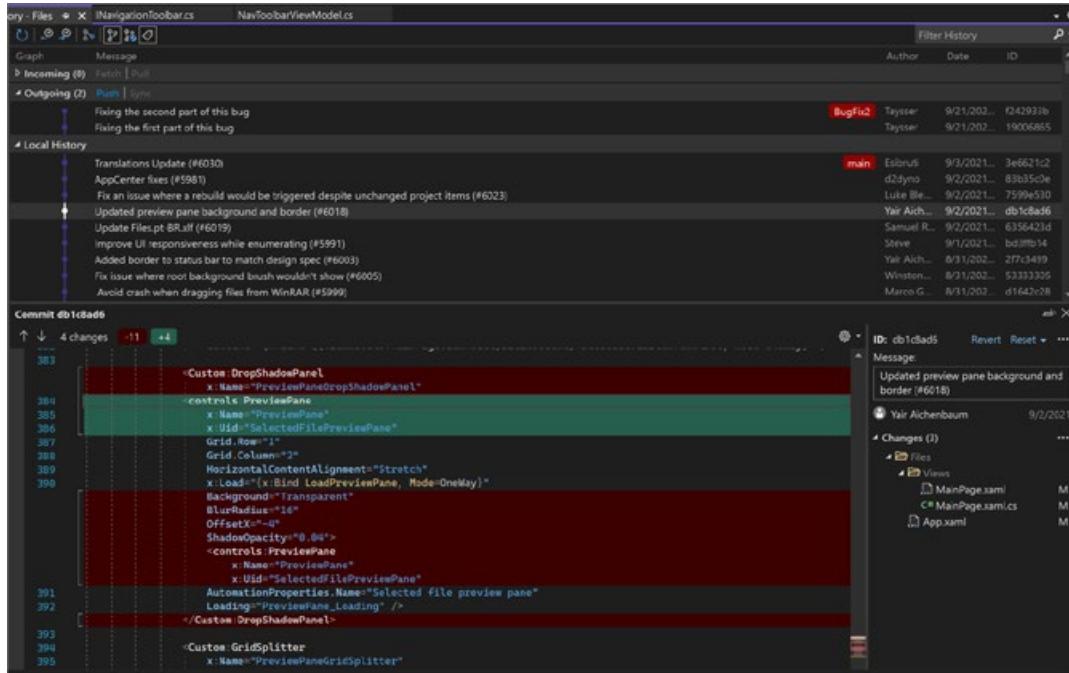
Microsoft Visual Studio Code：VS Code 内置了源代码控制集成，并且具有所有可用的扩展，您可以完全避免使用单独的程序。



Visual Studio 代码

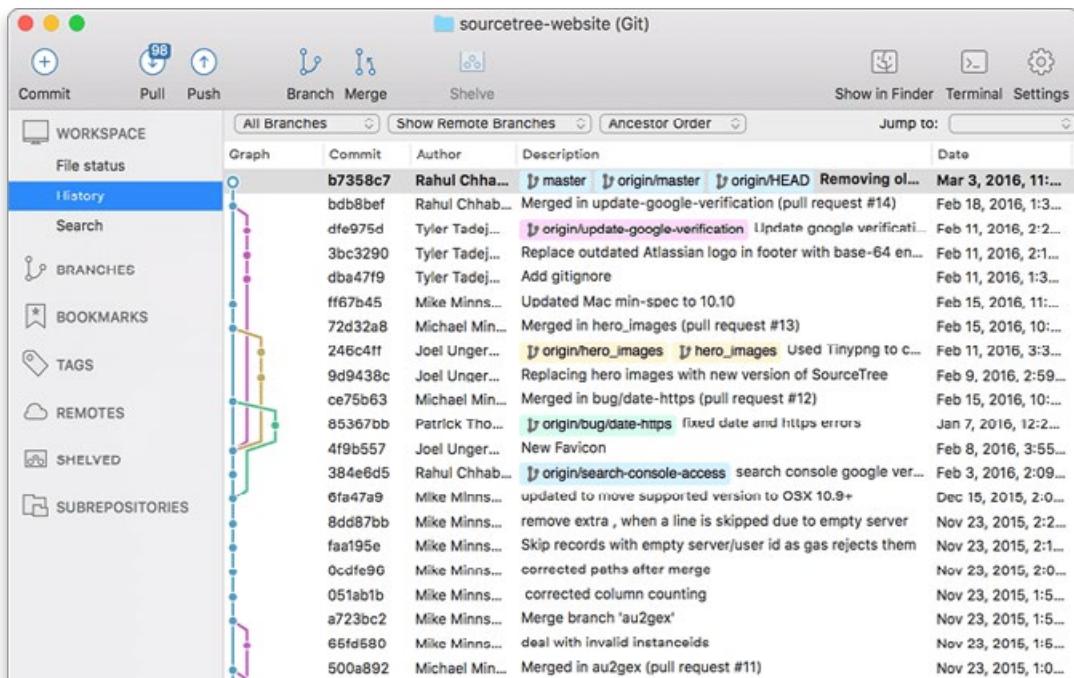


Microsoft Visual Studio: 与 VS Code 一样, Visual Studio 也内置了 Git 控件, 并包含一个 GitHub 扩展。



Visual Studio 的

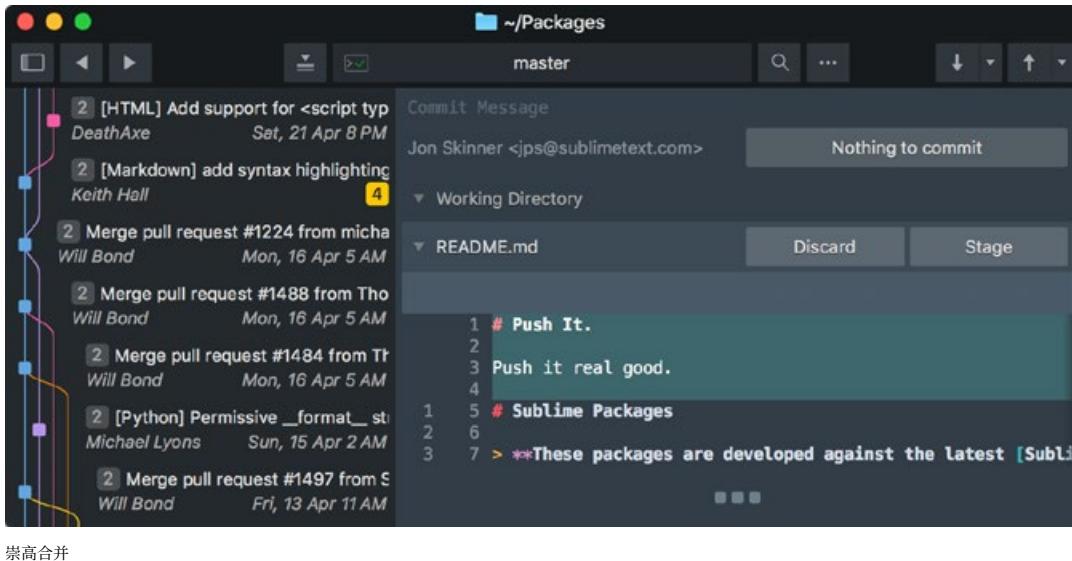
SourceTree: SourceTree 是 Atlassian 产品组的一部分, 是一款适用于 Windows 和 Mac 的免费 Git 客户端, 还可以帮助您轻松可视化和管理 Git 存储库。



SourceTree (源树)



Sublime Merge: 该系统提供了通过并排差异和语法突出显示来加速代码审查的工具。它是一个轻量级、高性能的客户端。



崇高合并

Git 通常被认为在分支和合并功能方面很强大，但它无法像市场上的其他解决方案那样有效地处理大型二进制文件。Git Large File Storage (LFS) 在一定程度上解决了这个问题。

由于 Git 是分布式客户端，因此整个存储库和完整历史记录都在开发人员的计算机上。这使得切换分支或恢复到历史记录中的某个时间点等操作变得非常快速。如果您正在处理具有多个功能和发布分支的大型项目，Git 工作流程可以节省无数小时。

Unity 已在 GitHub 上向公众发布了他们的 C# 编辑器和引擎代码，当您需要了解某些函数的工作原理或如何在自己的项目中复制编辑器的功能时，这非常有用。

GitHub 也有自己的 Git GUI，即 GitHub Desktop。在 Unity 中工作时，您还可以使用 GitHub for Unity 包将 Git 工具直接引入 Unity 编辑器。

如前所述，GitHub 并不是可用于 Git 项目的唯一托管服务。您还可以使用 Bitbucket (来自 Atlassian) 或 GitLab，它们有更多的 DevOps 功能可供他们使用，或者许多其他可用的托管服务之一。

请参阅 Unite Now 2020 的此演讲，了解如何开始使用 Github、GitKraken 和 Unity。



Perforce (螺旋核心)

Helix Core 是大型游戏工作室通常使用的企业级版本控制系统。这些工作室使用 Perforce，因为它具有集中式存储库，这些存储库通常托管在他们自己的服务器上。它没有可视化存储库，因此对于非技术开发人员来说，采用它可能更具挑战性，但在大型工作室中，将有 DevOps 和发布工程师帮助管理代码库。此外，作为企业解决方案，它包括一个全球支持团队。

Helix Core 也可以由小型团队使用，您仍然可以通过 Amazon AWS 或 Microsoft Azure. 等解决方案部署到云中

The screenshot shows the Helix Core P4V interface. The main window displays a changelist (ID 27) with various changes made by user 'Ben' on 17/11/2021 at 21:54. The changes include copying files from 'main' to 'dev', merging streams, and adding new files. A detailed view of a specific change (ID 27) is shown in a modal window, providing information about the file being modified ('AnotherScript.cs'), its revision ('rev20,22'), and its status ('Added in 1.0 release phase'). The interface includes standard Windows-style menus (File, Edit, Search, View, Actions, Connection, Tools, Window, Help) and toolbars.

Helix Core P4V 接口

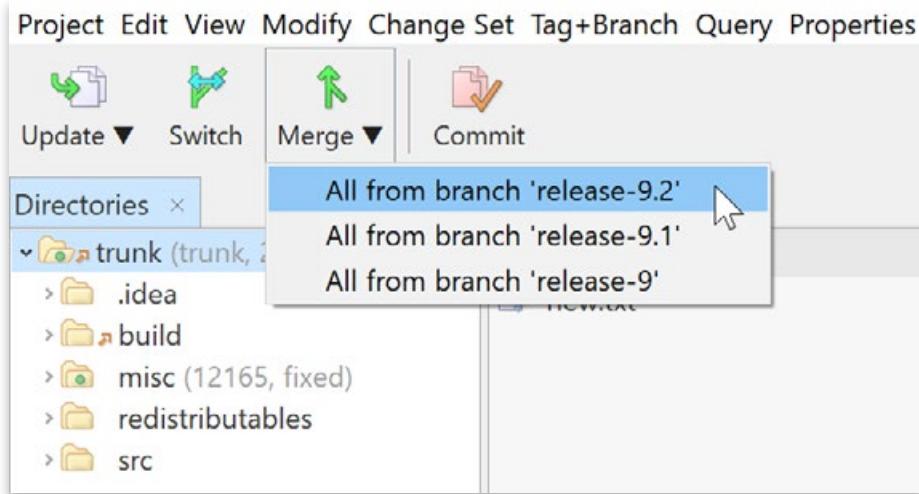
Helix Core 可以处理大型文件，并且有一个内置的 Unity 编辑器集成，将在后面的章节. 中介绍。

要了解有关将 Unity 工作流与 Helix Core 集成的更多信息，请查看此 Perforce 博客文章.



亚派克子版本

与 Git 一样，Apache Subversion（称为 SVN）是一个免费的开源版本控制系统。与 Git 不同，它是一个集中式 VCS，可以处理大型二进制文件。但是，它仍然是一个命令行系统，需要众多第三方 GUI 客户端之一更加用户友好。SmartSVN 就是这样一个客户端



SmartSVN 图形用户界面

在 Git LFS 之前，SVN 是在 Unity 中工作时的常见选择。作为集中式解决方案，它更易于使用，并且如前所述，更适合处理大文件。SVN 落后于其他工具的地方是当你开始使用分支并需要在它们之间进行合并时。在 SVN 中合并有很多痛苦，尤其是当涉及到文件之间的冲突 - 甚至是错误的冲突 - 时。在另一个 VCS 中需要几分钟的合并作可能需要数小时才能在 SVN 中手动完成。

有关设置 Unity 以使用 SVN 的更多信息，请查看 Unity 文档。

Unity 版本控制

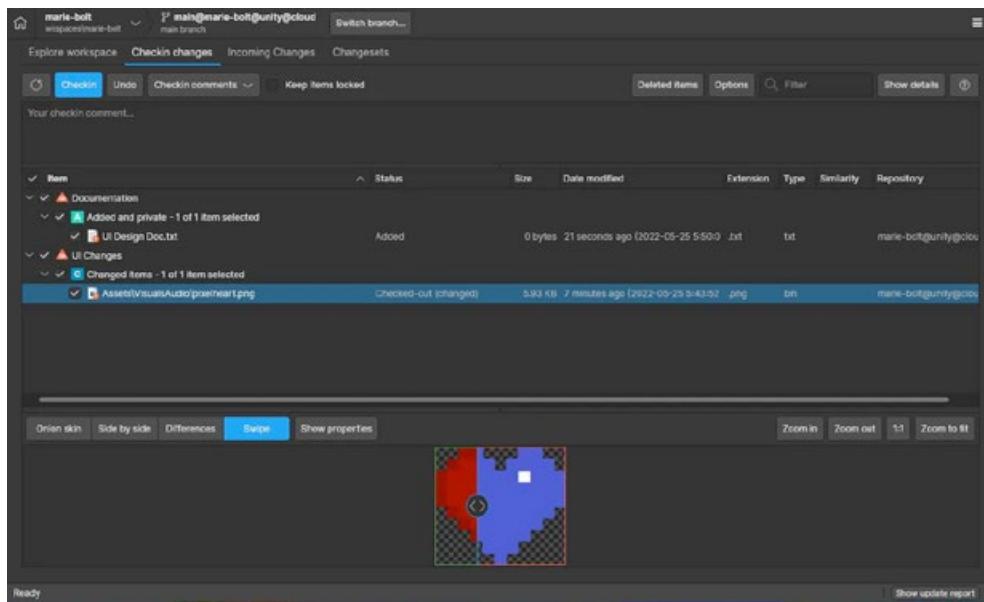
Unity 版本控制（UVCS）是一个灵活的版本控制系统，具有独特的界面，可为程序员和艺术家提供支持。它擅长处理大型存储库和二进制文件，并且作为基于文件和基于变更集的解决方案，它使您能够仅下载您正在处理的特定文件，而不是整个项目构建。

有三种方法可以访问 UVCS：通过 UVCS 桌面客户端通过多个应用程序和存储库，通过 Unity Hub 将其添加到项目中，或通过 Web 浏览器访问 Unity 云上的存储库。有关如何设置的更多信息，请参阅“Unity 6 中的 UVCS 入门”部分。

最多三个用户可以免费注册，在撰写本文时，最多可获得 5GB 的云存储空间，以及版本控制软件的访问权限，包括 Gluon。

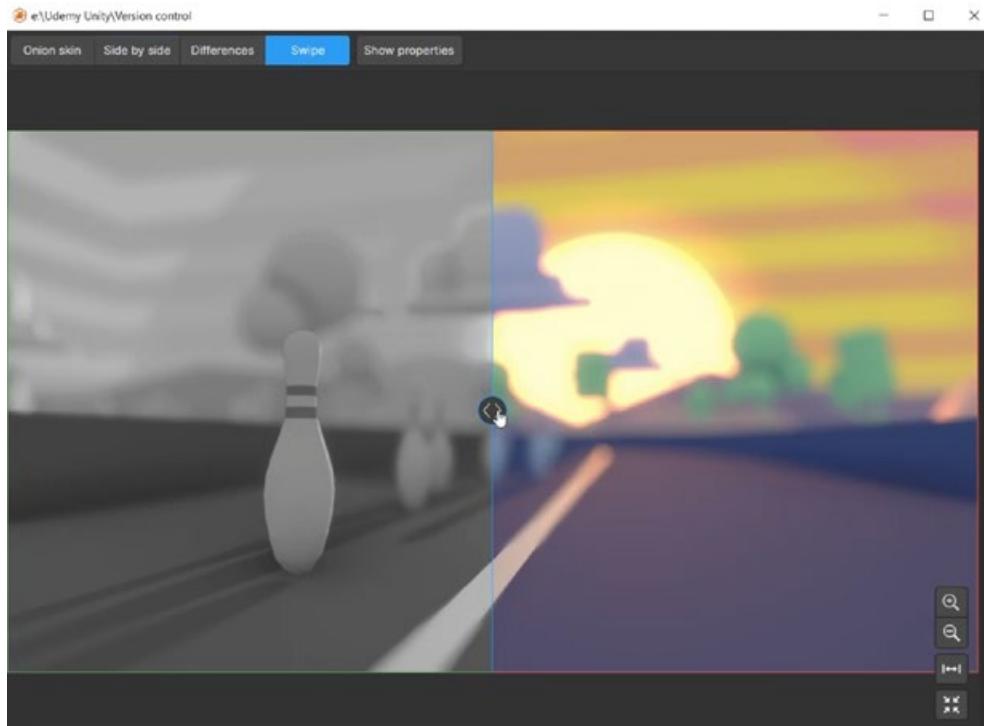


Gluon 是一个精简的客户端，旨在让艺术家像艺术家一样工作，而不是程序员。它允许您只选择要处理的文件并从服务器中签出它们，从而锁定它们不被其他任何人修改。完成工作后，将文件签回。Gluon GUI 删除了更复杂的概念，这些概念更适合程序员，而不是其他技术含量较低的用户。



Gluon 提供了专为艺术家设计的工作流程，使预览文件和历史记录以及签入更改变得容易。

对于艺术家，UVCS 和 Gluon 都包含对图像进行比较的方法。图像比较工具允许您直观地比较同一文件的两个版本，这是许多其他系统不提供的功能。

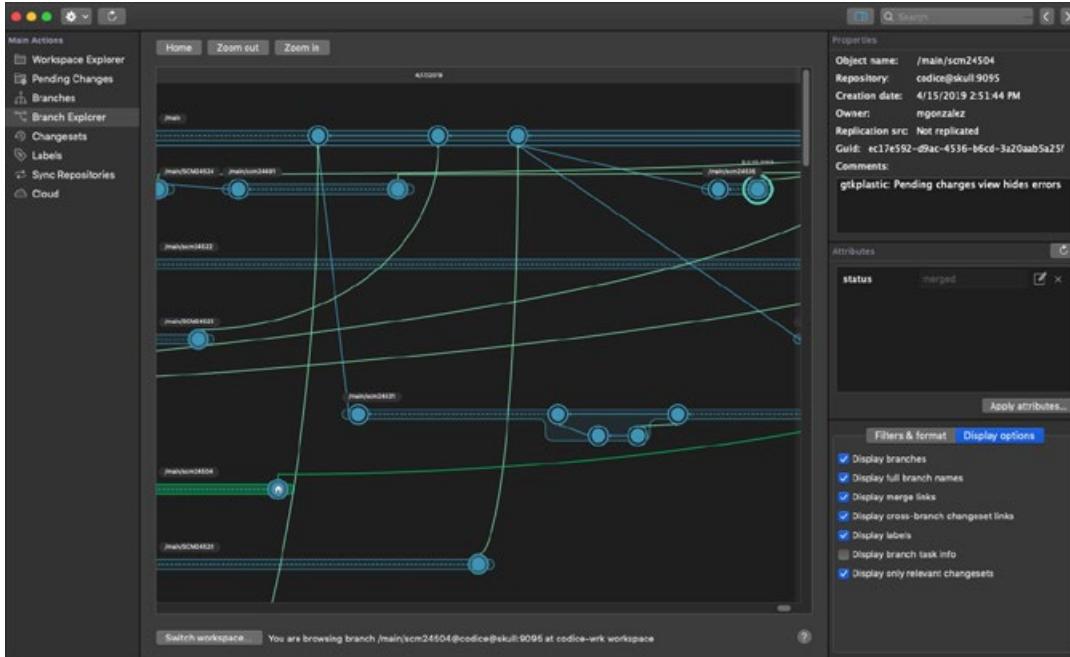


Swipe 中的差异查看器 mode：通过拖动滑动控制从一个版本转到另一个版本，这是一个有用的功能。

R 跟踪图像演变



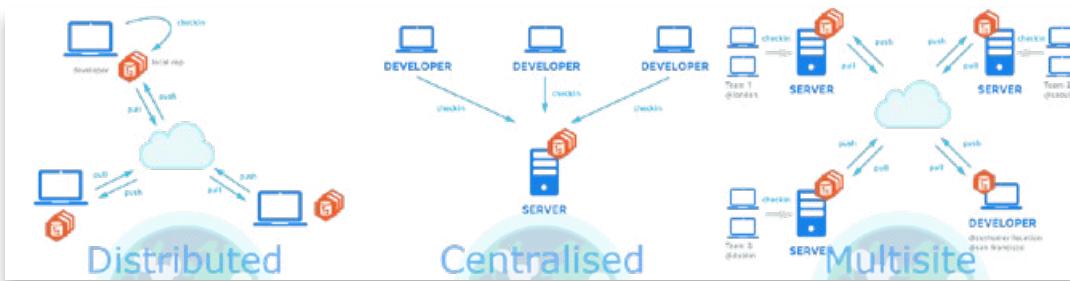
标准 UVCS GUI 客户端具有他们正在寻找的所有功能，并为编程团队提供更多功能。GUI 具有交互式可视化 Branch Explorer，可显示项目中所有分支的真实关系。还有一个内置的代码审查系统，您可以使用它来请求其他组或团队成员对您的工作进行审查。



分支资源管理器可视化项目的合并结构。它从左到右水平演变。

UVCS 的主要优势之一是它可以灵活地配置为分布式或集中式工作流程。在完全分布式模式下，开发人员在本地计算机上使用存储库，轻松签入、分支和合并。然后，开发人员会将更改推送和拉取到服务器，以便在准备就绪时共享它们。

在集中式模式下，用户将他们的更改直接签入并签入服务器，以便每个人都在处理最新的更改。然而，随着开发团队发展成为全球性组织，每个人都与一台中央服务器通信并不总是有益的。UVCS 也可以配置为在多站点系统中工作。在这个系统中，服务器在每个站点都设置好，因此团队可以签入他们的本地服务器，保持他们的工作流程快速和硬盘驱动器满意。然后，分布式服务器通过中央或云服务器相互通信。



有关在 Unity 中设置 UVCS 的步骤，请参阅 Unity 6 中的 UVCS 入门部分以及此 Unity Learn 快速入门教程。



VCS 比较

		UVCS	Git	Perforce	Subversion
Flexibility	Work centralized Checkin only, no push/pull				
	Work distributed Push/pull + local repo				
Binaries	Large repos				
	Large files				
	Lock files to avoid merging				
GUI	Visualizes your repos (so you don't need a PhD in branching)				
	User-friendly GUIs				
	Artist-friendly GUI and workflow				
Workflow	Creates effective task branches				
Merge	Detects merges between branches				
	Provides diff and three-way merge tools				
	Tools help you understand the merge				
	Good merging renames, moved files, directories, refactors				



Cloud	Can host repos in the cloud				
	Cloud hosting is good with large repos				
DIFF	Can diff code moved across files				
	Shows you the history of a method				
	Enterprise Support				

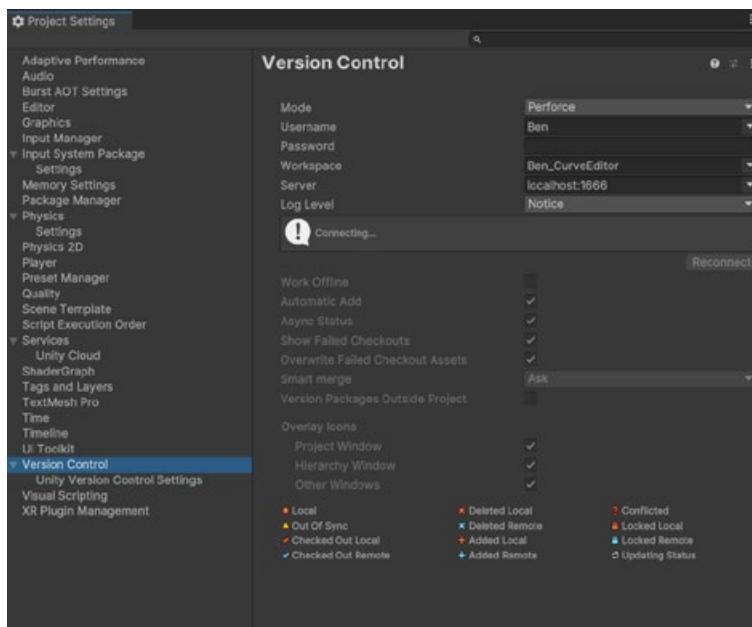
设置 Unity 以使用版本控制

本部分将帮助您根据自己的偏好设置 Unity 以使用 Git、Perforce 或 UVCS。通过了解每个解决方案的一些关键工作流程，您可以就哪种系统最适合您的团队做出明智的决定。

Editor 工程设置

Perforce 融合核心

大多数版本控制系统都提供 Unity 编辑器集成，并且 Perforce Helix Core 集成内置于编辑器中。您只需通过编辑 > 项目设置 > 版本控制. 将模式设置为 Perforce，并填写工作区和服务器设置的信息.

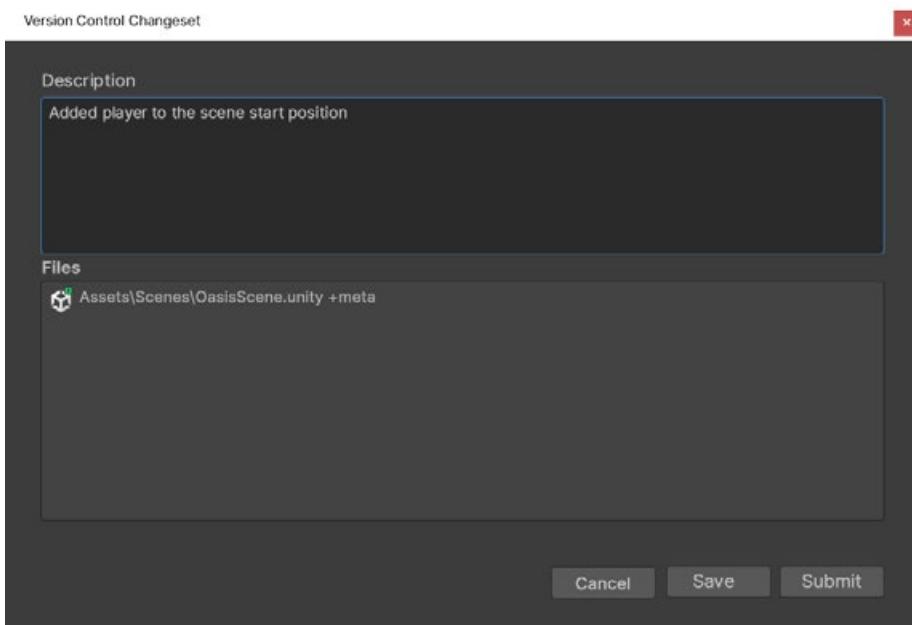


为项目设置 Perforce Helix Control



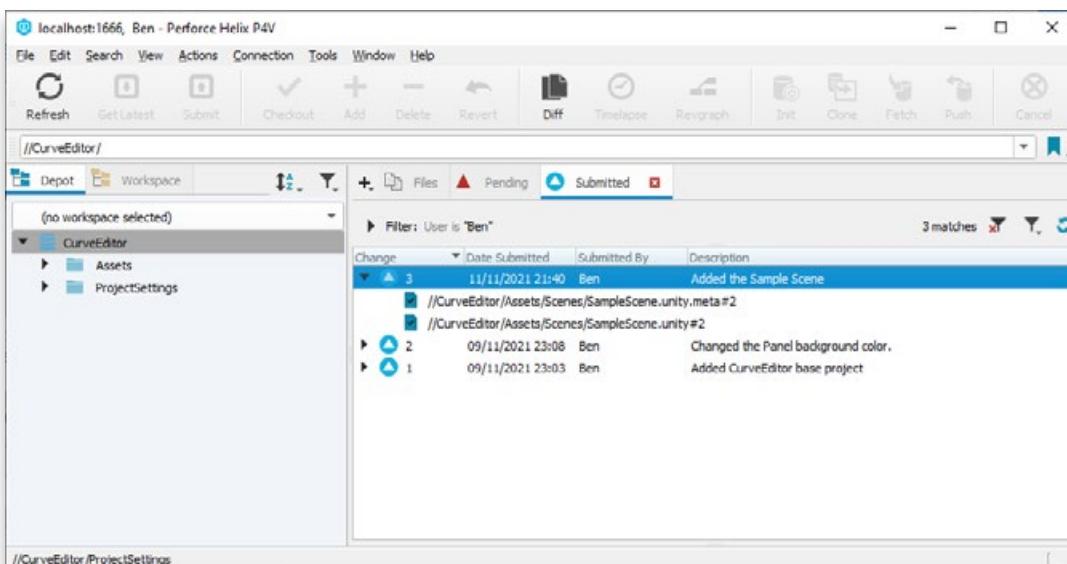
启用此功能后，您将看到文件现在被视为“Under Version Control”，并有签出选项。

签出文件后，您可以锁定、解锁、提交或还原文件。选择 submit 将弹出一个变更集对话框，供您在将提交消息提交到存储库之前添加提交消息。



“变更集”对话框

使用 Helix P4V 界面查看项目历史记录。



查看项目历史记录。

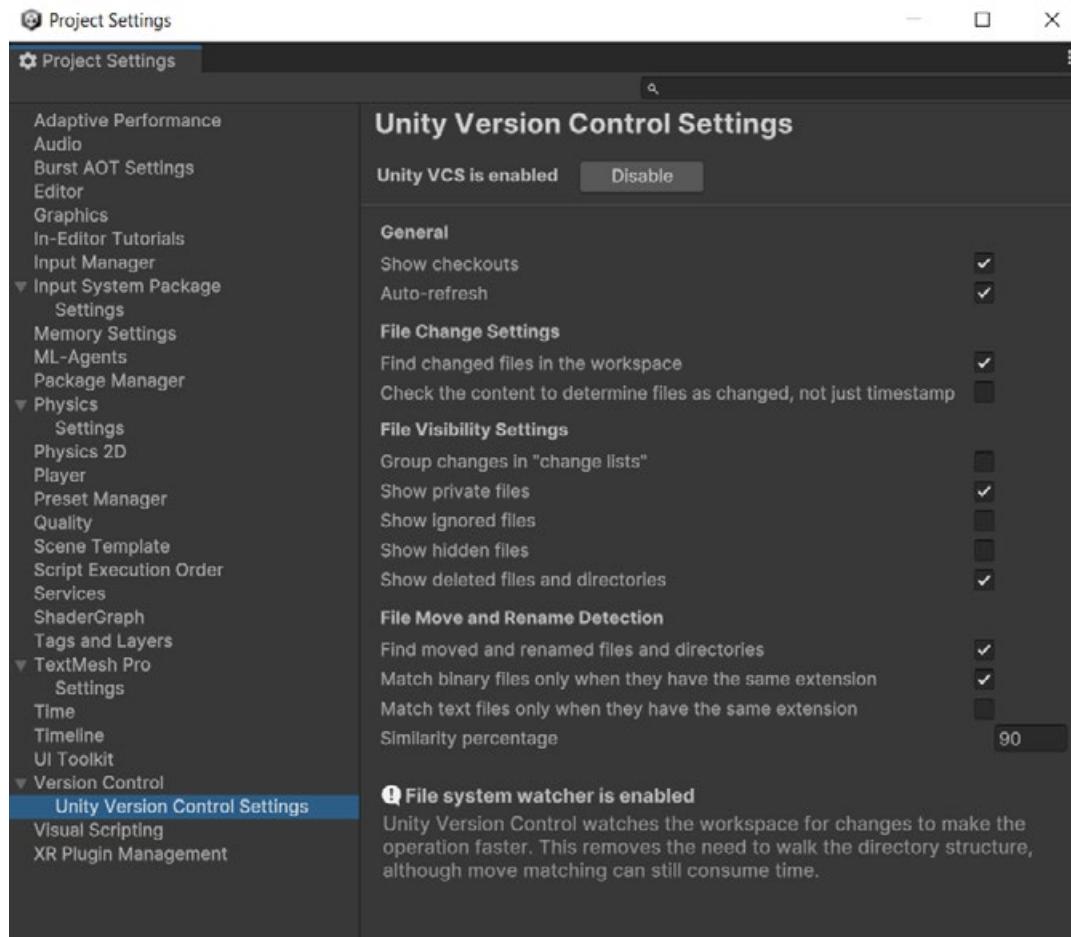
有关开始使用 Perforce Helix Core 和 Unity 的更多信息，请查看 Perforce 博客。



UVCS 抗体

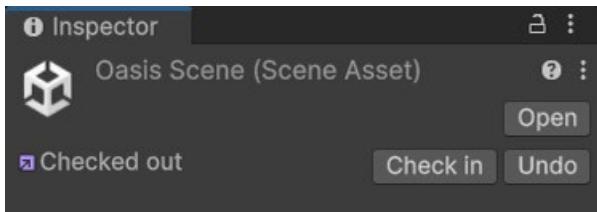
UVCS 与 Unity 编辑器集成，便于入门和简化工作流程。在 Hub 中创建新的 Unity 项目时，可以通过选中 Use Unity Version Control 复选框立即集成 Unity VCS。

要将 UVCS 连接到 Editor 中的现有项目，请打开 Window > Unity Version Control。该选项卡将显示在 Project 窗口中。

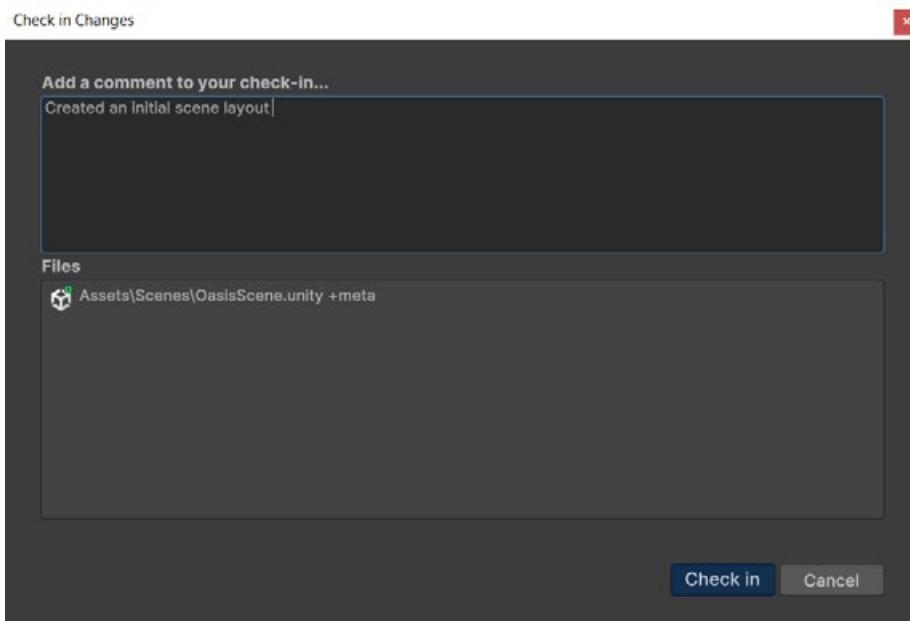


Project Settings 中的 UVCS

UVCS 对 Unity 用户来说会很熟悉，并且无需额外的客户端，从而简化了工作流程。可以直接从 Editor 添加、签出、还原、签入或提交文件。

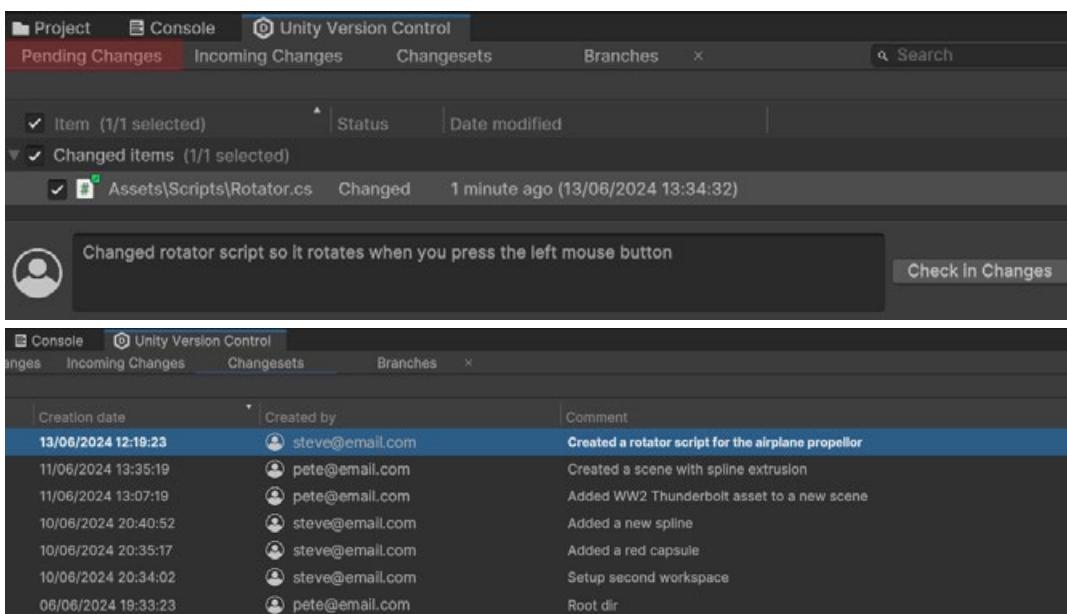


从 Unity Editor 处理 UVCS 中的文件



签入文件

UVCS 还有一个优势，即通过窗口 > 在 Unity 编辑器中提供 Changesets 选项卡 Unity Version Control.



Pending changes 和 Changesets 选项卡



有关在 Unity 中设置版本控制的更多信息，请查看文档。

Git 和其他解决方案

对于所有其他版本控制系统，打开 Edit > Project Settings > Version Control 窗口，然后从下拉菜单中选择 Visible Meta Files。这里没有其他选项，但元文件必须可见，版本控制系统才能检测到它们（参见上一章关于 Meta Files）。

要忽略的内容

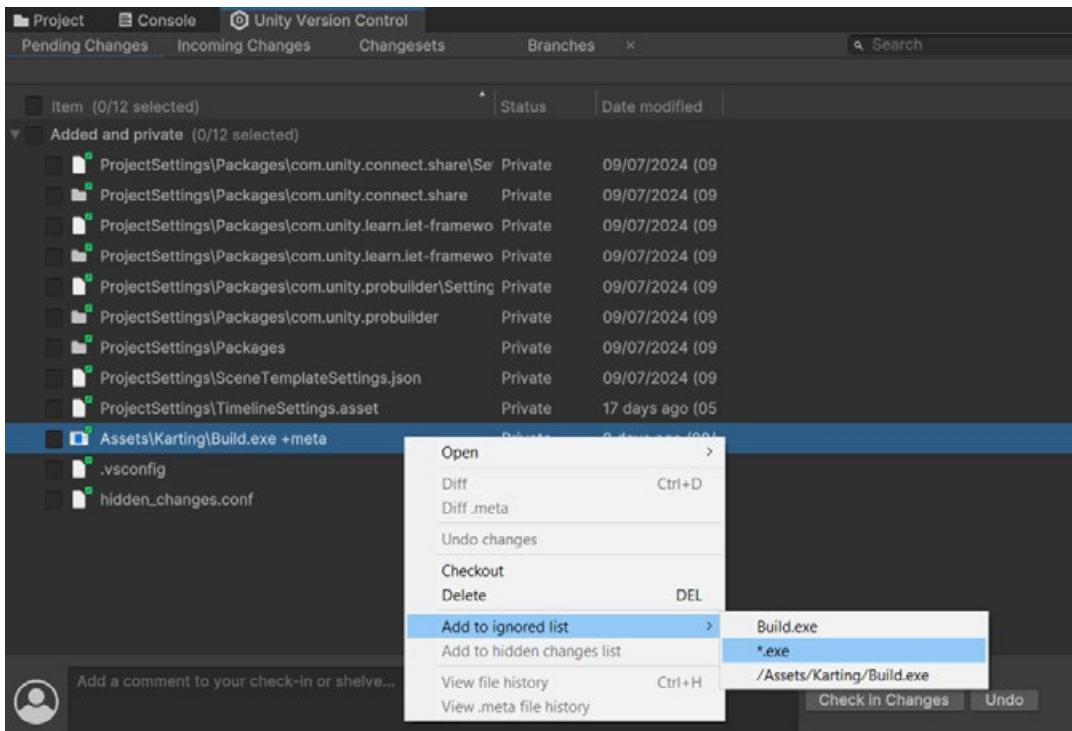
使用 Unity 项目或任何相关项目时，应仅将无法生成的文件置于版本控制之下。

对于 Unity 项目，这意味着只应将 Assets 和 Project Settings 文件夹中的文件提交到存储库。Unity 可以自动重新创建所有其他文件夹。在任何情况下都不应提交 Library 文件夹，因为此文件夹可能会变得非常大，如果不存在，Unity 将在启动 Editor 时重新创建它。

— 从 Unity 编辑器进行设置时，UVCS 会自动选择适当的文件夹和文件以置于版本控制之下。有一个列表保存在项目根目录下的 ignore.conf 文件中，用于描述要忽略哪些文件。要了解有关设置“ignore.conf”文件的更多信息，请查看此博客文章。
— 使用 Perforce，您需要将 Assets 和 Project Settings 文件夹显式添加到您的 depot。
— Git 需要一个 .gitignore 文件来指示哪些文件不应包含在内。根据您的 Git GUI 客户端，您可以在创建存储库时选择模板，或者如果您先设置托管，可以通过 GitHub 完成此操作。或者，也可以在此处下载模板。

您还应该避免提交 .exe 或 .apk 文件等内容。此外，不应将从您的 Unity 项目构建的 gradle 和 xcode 项目添加到存储库中。

此规则的一个小例外是，如果您要为 Gradle 或 Xcode 项目设置自动化构建流程，但随后它们通常会提交到自己的存储库。



使用 UVCS 时，可以直接从 Unity Editor 将文件添加到忽略列表中。

使用大型文件

Unity 项目由许多代码组成。事实上，在 Unity 项目中，脚本的数量通常远远超过其他资源文件。这些资源存储为二进制文件：纹理、模型、预制件、音频剪辑、时间轴等。这会导致两件事：

- 它们可能很难在修订版之间进行比较。
- 无法描述 diff，因此在将更改推送到存储库时写入整个文件。

同样，在分布式环境中，整个项目历史记录在用户的本地计算机上可用。现在，如果您有一个大文件的历史记录，这些文件在很长一段时间内发生了许多更改，那么您的计算机上将存储该文件的多个副本。这会很快占用您的大部分硬盘空间！

这是历史上团队更喜欢集中式工作流程的主要原因之一。这样，二进制文件的大型历史版本将仅存在于中央服务器上，单个用户只能在其计算机上访问最新版本。

Perforce 和 UVCS 都是集中式系统，可以很好地处理大文件。UVCS 还为您提供在分布式管道中工作的选项，而大文件大小是您在这些选项之间进行选择时需要考虑的权衡。



UVCS 的另一个功能是 Dynamic Workspace，它依赖于虚拟文件系统。这意味着 Dynamic Workspace 按需下载文件 – 因此，虽然您可以看到工作区中的所有内容，但实际上并非所有内容都已下载。

分发的 Git 可能会难以处理大文件。如果您要处理大文件，请务必还包括 Git LFS。Git LFS 将 .git 文件夹中的大文件替换为文本指针，同时将实际资产存储在 GitHub. 等服务器上

版本控制的最佳实践

无论您使用哪种 VCS，许多最佳实践都可以帮助您的团队更有效地工作。每个团队都有不同的需求，因此每次练习都不适合每个团队。

这些提示来自 Unity 企业支持团队，他们正在帮助一些最大的工作室优化实际项目。

少投入，经常投入

这是迄今为止您可以对工作流程进行的最简单的更改，但也是一些开发人员最难以应对的更改。使用其他项目管理工具时，您可能已经将工作分解为小的、可管理的任务。提交应完全相同。

单个提交应该只与一个任务或工单相关，除非一行代码神奇地修复了几个错误。如果你正在开发一个更大的功能，请将其分解为较小的任务，并为这些任务进行提交。我们稍后将深入研究功能分支。

使用较小提交的最大优点是，当出现问题时，你将更容易找到更改，并且可以在不影响任何其他积极更改的情况下还原负面更改。



保持提交消息干净

提交消息描述了您的项目的历史。如果游戏的提交消息显示“Added high score tables to the menu”而不是“Menu updated”，那么找到为游戏添加高分表的更改要容易得多。

当使用 JIRA 或 GitLab 等任务工单系统时，最好在提交中包含工单编号。许多系统可以设置为与智能提交一起工作，在这些系统中，您实际上可以引用工单并从提交消息中更改它们的状态。

例如，提交“JRA-123 #close #comment task completed”会将 JIRA 工单 JRA-123 设置为 closed，在工单上留下注释“task completed”。

有关设置此工作流的更多信息，请参阅 JIRA 中的文档或 GitLab 中的 Pivotal Tracker 服务

避免不分青红皂白的提交

唯一应该使用“commit -a”（用于“commit all changes”的 git 命令）或其任何对应项的时间是项目的第一次提交。通常，这是项目中唯一 README.md 文件时。

提交应仅包含与你提交到存储库的更改相关的文件。使用 Unity 项目时应特别小心，某些更改可能会导致多个文件被标记为已更改，例如场景、预制件或 sprite 图集，即使您不打算对它们进行任何更改。

如果您不小心将更改提交到其他人正在处理的场景，这可能会让他们头疼，因为他们去提交更改并发现他们需要先合并您的更改。

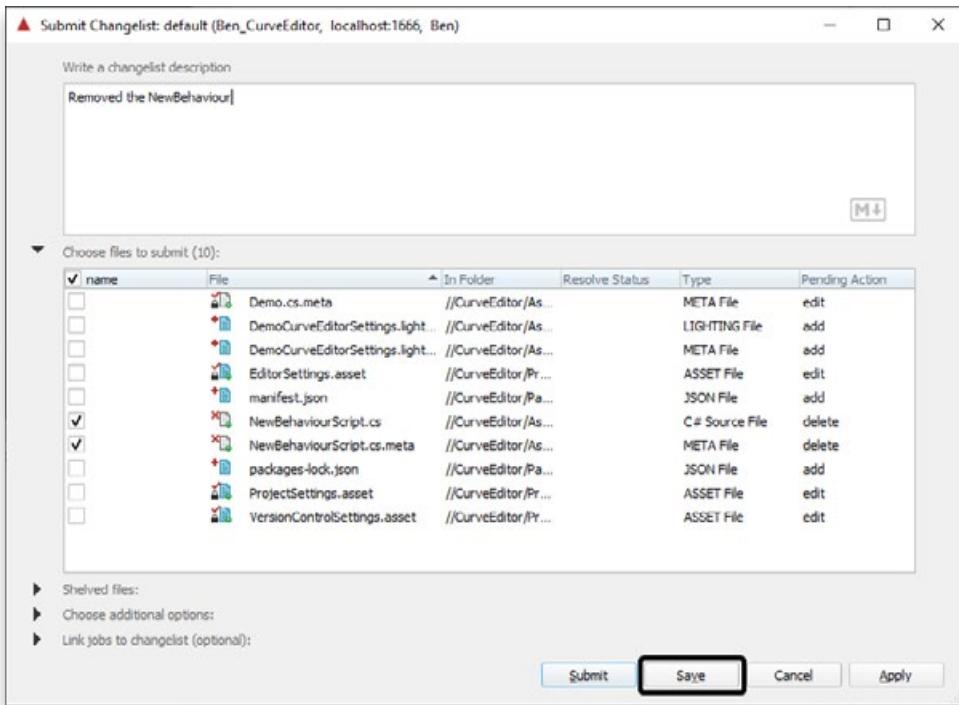
这是版本控制新手最常犯的错误之一。重要的是要了解您应该只提交项目中 *you* 更改的内容。要了解更多信息，请查看这篇关于如何加快工作流程的博客文章。



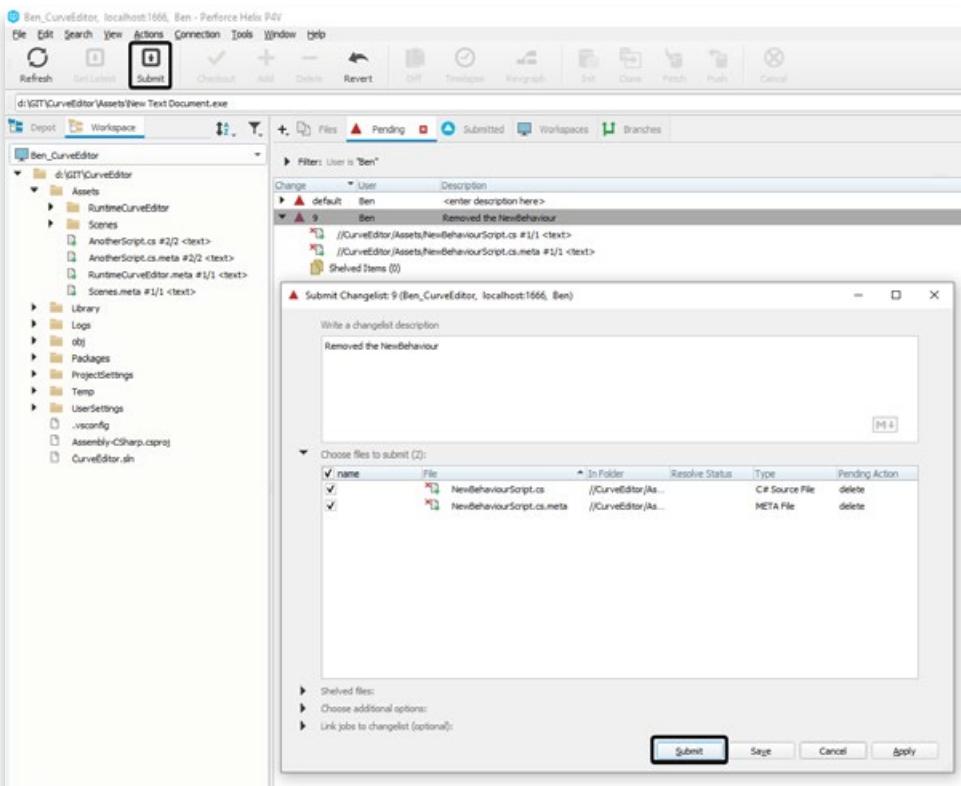
获取最新版本

只要有意义，就把最新的更改从 repo 拉到你的工作副本中。孤立地工作是不好的，因为这只会增加合并冲突的可能性。每个系统中典型的日常工作流程是这样的。

Git	Perforce
<ul style="list-style-type: none">— git pull— Then as many times as you like:<ul style="list-style-type: none">— Make edits in your working copy.— git commit your changes.— git pull the latest changes.— Once you are happy with your changeset of commits:<ul style="list-style-type: none">— git pull once more.— git push to send your commits to the repo.	<ul style="list-style-type: none">— Get latest— Check out files to work on— Make edits— Submit changes



在 P4V 中保存对新变更列表的更改



在 P4V 中提交变更集

UVCS 工作流程略有不同，因为您可以在集中式、分布式或多站点配置中工作。

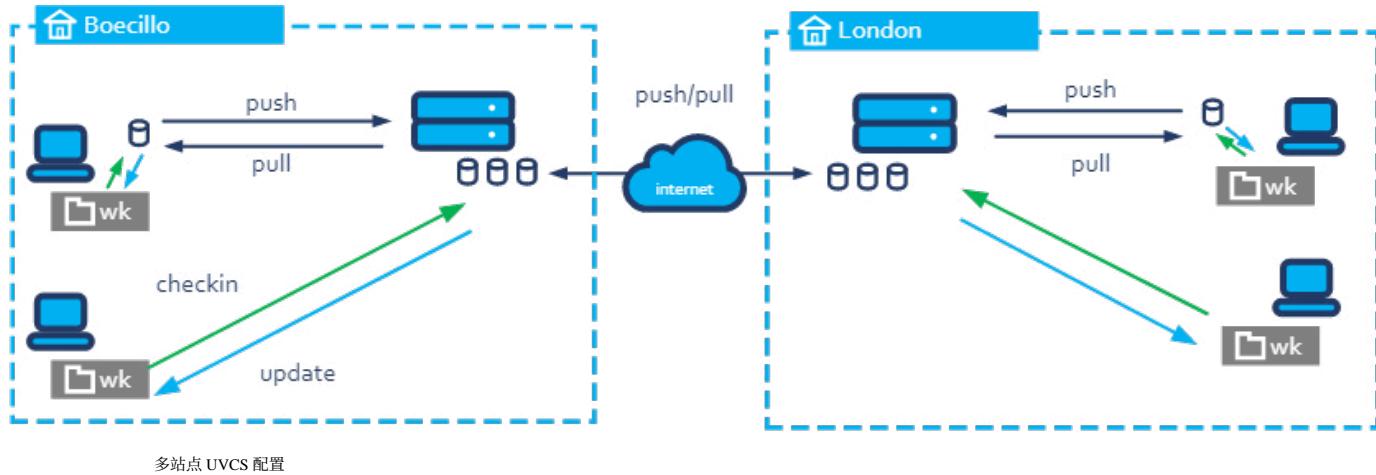
UVCS (centralized)	UVCS (distributed)	UVCS (multi-site)
<ul style="list-style-type: none">— Sync repositories— Pull visible— Check out files to work on— Make edits— Check in changes— Sync repositories— Push visible	<ul style="list-style-type: none">— Pull changes from the server— Check in changes to your local copy— Pull any new changes— Push your changes back up to the server	<ul style="list-style-type: none">— A hybrid of the two, depending on your setup

多站点配置可以根据自定义需求进行定制，每个用户在集中式或分布式工作流程中工作。



请考虑以下两个团队的示例：

- 每个团队都有一个现场服务器。
- 两个站点的团队成员在本地或分布式签到，但受益于现场关闭服务器的速度。
- 服务器在彼此之间推送/拉取以保持完全或部分同步。

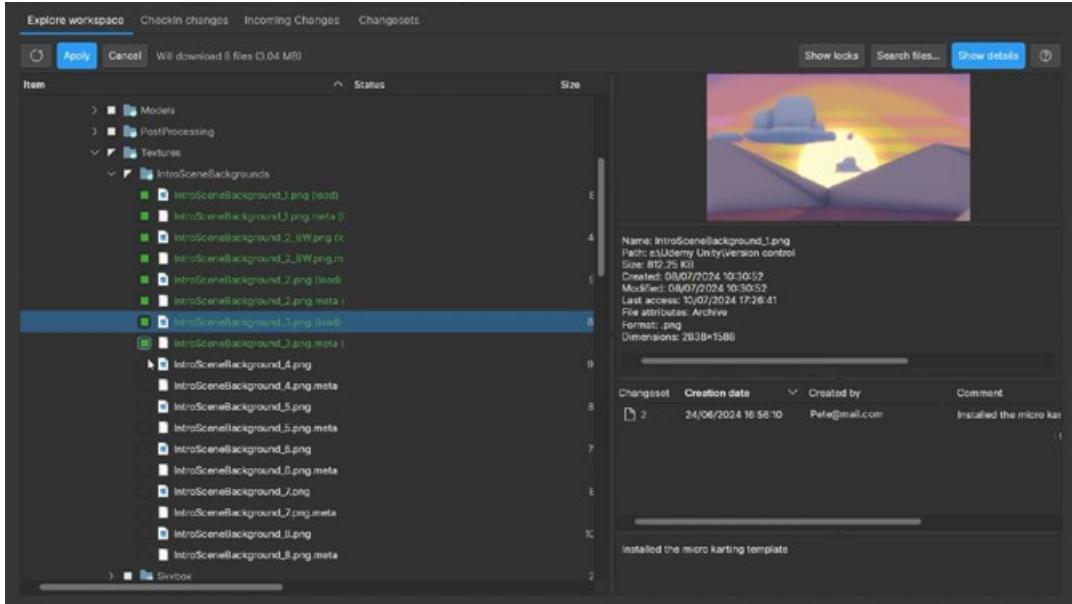


了解您的工具集

无论您的团队选择使用哪种 VCS，请确保团队能够舒适地使用它并了解他们可以使用的工具，包括视觉客户端。

如果您正在使用 Git，则并非每个人都需要使用相同的 GUI 客户端。但请确保每个人都熟悉提交 > 拉取 > 推送工作流程，并且他们知道如何仅提交他们需要的文件。

如果您正在使用 UVCS，请让您的艺术家熟悉使用 Gluon 来简化他们的工作流程。Gluon 允许您决定要处理哪些文件，并且只下载这些文件，无需下载和管理整个项目。它允许您锁定文件以防止其他人处理该文件，并且一旦您完成，用户可以将文件提交回存储库并再次解锁它们。

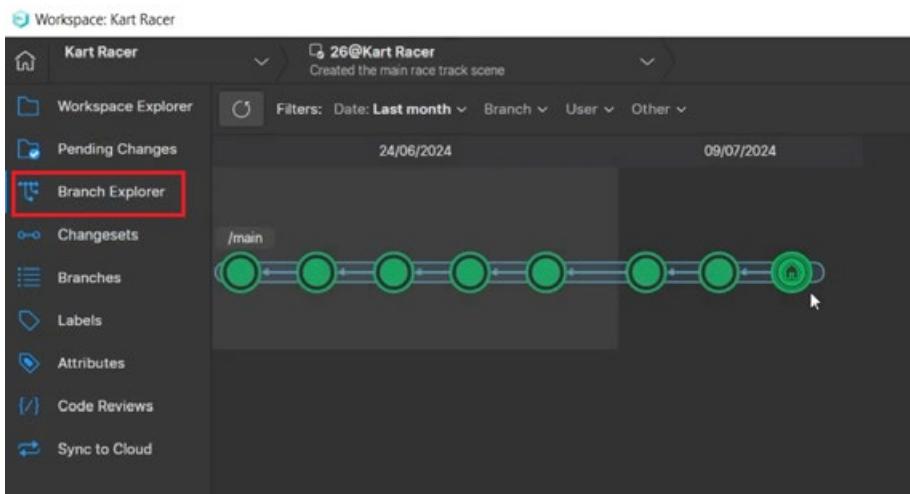


UVCS 中的胶子

如果您使用的是 Perforce Helix Core，请使用内置的 Unity 编辑器工具直接从编辑器管理版本控制。这对于艺术家或 Unity 资源文件（如场景、预制件等）的一般处理都非常有用。您可以在 Editor 中签出要修改的资源，进行更改，然后重新签入它们，甚至无需离开 Unity。

功能分支和 Git 流

当您处理具有多个发布周期的长期项目时，功能分支对您的工作流程非常有益。通常，团队在存储库的同一分支中工作，该分支可能称为 trunk、master 或 main。执行此操作时，整个项目将沿同一时间轴移动。

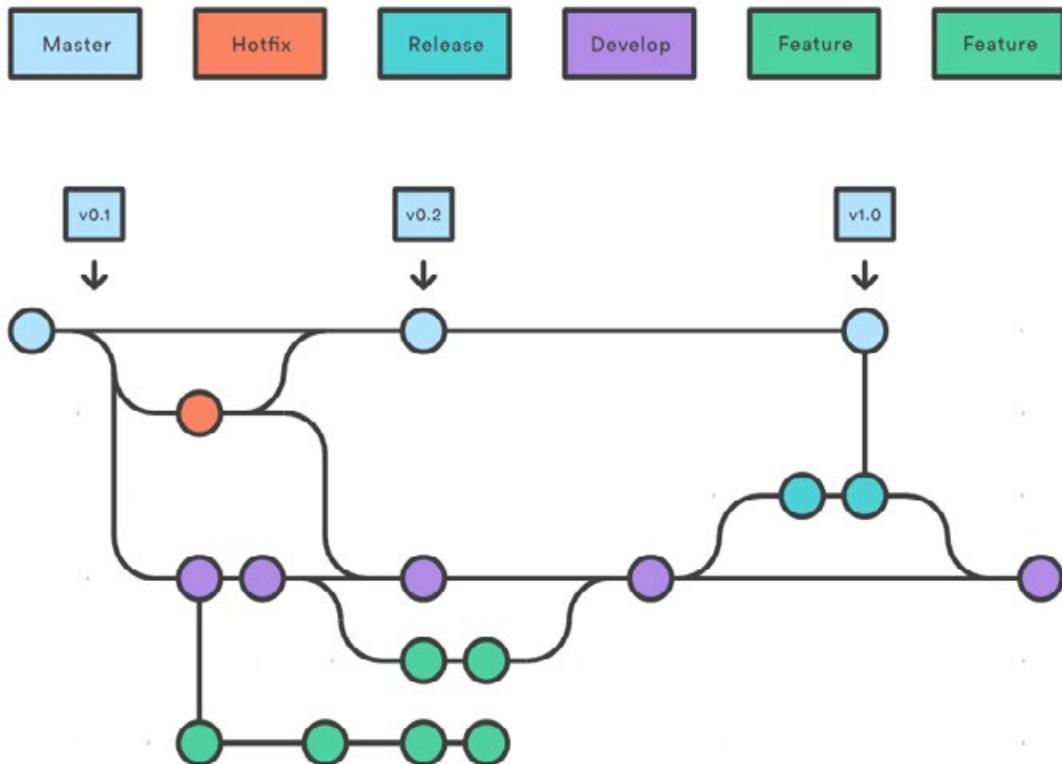


UVCS 中沿主分支的发展



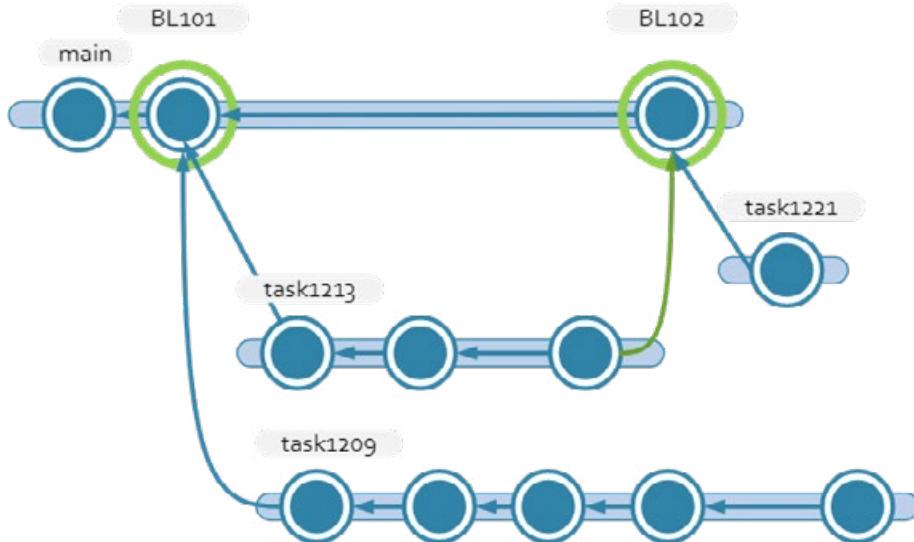
但是，将工作拆分为多个分支以更有效地作为一个团队工作可能是有益的。

在 Git 中，一个名为 Git Flow 的特定工作流专注于使用不同的分支来提供功能、错误修复和发布。开发人员在独立分支中开始开发新功能，完成后，它会合并回主分支。同时，其他人可能不得不对以前的版本进行热修复，修复一个错误，并安全地发布新版本，而不包含任何仍在开发中的功能。



Git Flow 工作流可以更轻松地进行发布管理。

UVCS 还具有任务分支。对于此模式，您将为跟踪的每个任务创建一个新分支。在 Git Flow 中，我们使用功能分支来开发完整的（有时是大型的）功能，而 UVCS 中的任务分支是短暂的。如果一个任务需要大量的提交来实现，那么它很可能被分解成更小的任务。

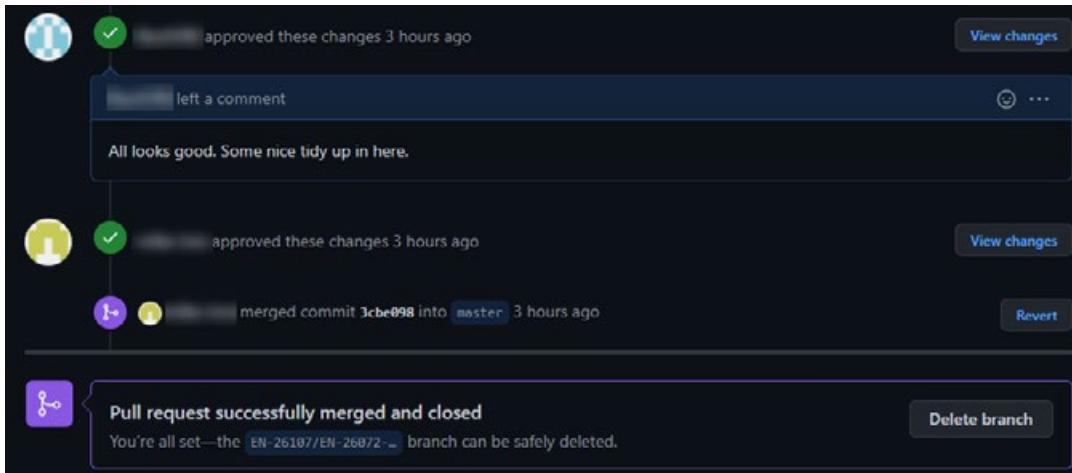


每个任务的 UVCS 分支模式

Perforce Helix Core 使用一个名为 Streams 的系统来促进这种风格的工作流程。创建要使用的站点时，需要将其设置为河流站点类型。然后，您可以使用 Stream Graph（流图）视图创建新流。除主线流之外的每个流都需要有一个父流，以便可以将更改复制回上游。

拉取请求

完成功能分支的工作后，最好使用拉取请求将更改重新纳入存储库的主流。拉取请求由功能或任务的开发人员创建，通常高级开发人员或 DevOps 有责任在将更改接受到主线之前对其进行审查。



GitHub 上的已关闭拉取请求



UVCS 和 Perforce 都有自动化工具来帮助管理将分支合并回主线。UVCS 在 Mergebot 的帮助下实现了这一点，一旦 Mergebot 的分支经过审查并通过验证，它就会自动合并仓库的分支。Perforce 还有一个额外的平台 Helix Swarm，用于管理代码审查，也可以通过自动化测试进行设置。

Review of changeset 37 - Added the nitro code to the Karting script

Review of changeset 37 - Added the nitro code to the Karting script Under review

Changeset 37 by Pete@mail.com

Added the nitro code to the Karting script

Path: Changed: 3 items

- /Assets/Karting/Scripts/AI/KartAgent.cs
- /Assets/Karting/Scripts/KartSystems/ArcadeKart.cs
- /Assets/Karting/Scripts/KartSystems/Inputs/KeyboardInput.cs

Semantic Outline

Text diff Semantic diff Visual diff

Current: 3/3

Comments for line 459

Request a change Ask a question

Change this to activate only if holding the left control button and not just when reaching max speed.

Comment Cancel

```
451     Vector3 fwd = turnAngle * tra
452     Vector3 movement = ((m_HasCol
453
454     // forward movement
455     bool wasOverMaxSpeed = curren
456
457     // If over max speed, cannot
458     if (wasOverMaxSpeed && !isBrai
459     movement *= 0.6f;
460
461     Vector3 newVelocity = Rigidbo
462     newVelocity.y = Rigidbody.lim
463
464     // clamp max speed if we are
465     if (GroundPercent > 0.0f && !i
466     {
467         newVelocity = Vector3.Clar
468     }
469
470     // coasting is when we aren't
471     if (Mathf.Abs(accelInput) < k_
472     {
```

453 Vector3 fwd = turnAngle * tra
454 Vector3 movement = ((m_HasCol
455
456 // forward movement
457 bool wasOverMaxSpeed = curren
458
459 if (wasOverMaxSpeed)
460 {
461 if (nitroAmount > 0.0f)
462 {
463 nitroAmount -= 0.1f;
464 movement += nitro;
465 }
466 }
467 else
468 {
469 // If over max speed,
470 if (wasOverMaxSpeed &&
471 movement *= 0.6f;
472 }
473
474 Vector3 newVelocity = Rig

UVCS 代码审查包含在 GUI 中。

Unity 6 中的 UVCS 入门

Unity DevOps 产品包括 UVCS 和 Build Automation。UVCS 对最多三（3）名团队成员（席位）免费，每月最多可使用 5GB 数据。之后，定价取决于您的每月活跃用户和总云存储。当您达到任何 UVCS 服务的使用限额的 50%、75% 和 90% 时，UVCS 会向 UVCS 组织所有者发送警告电子邮件，以便您轻松跟踪您的使用情况。

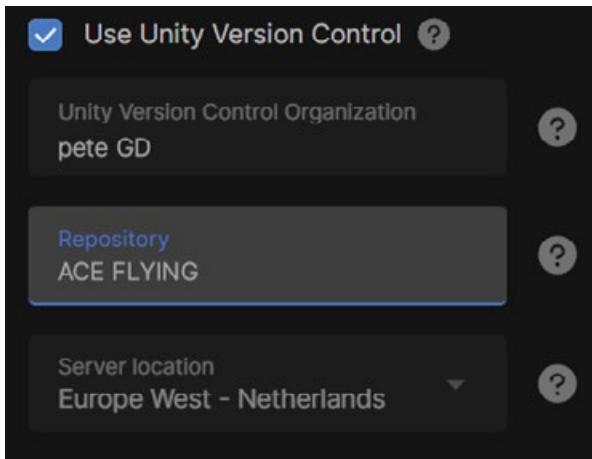
有关更多信息，请参阅 Unity Cloud 定价计划页面，或者登录 Unity Cloud 控制面板并转到 DevOps 控制面板的“关于”页面。

有三种方法可以访问 UVCS：通过 UVCS 桌面客户端通过多个应用程序和存储库，通过 Unity Hub 将其添加到项目中，或通过 Web 浏览器访问 Unity 云上的存储库。



在 Unity 项目中使用 UVCS

当您使用 Unity Hub 创建项目时，Unity 将自动设置存储库（repo）并选中 Version Control 复选框。



在 Unity Hub 中设置版本控制

这意味着您的项目数据和文件将既存储在本地计算机上，也可以存储在云上，以帮助协作并充当安全备份。您的项目名称将成为存储库的名称，您可以选择离您最近的服务器进行高速连接。

Assets、Packages 和 Project Settings 文件夹在首次使用时备份到服务器上。您可以通过登录 Unity Cloud 并在存储库中选择 File Explorer 来访问这些文件。这将显示 Cloud 存储库中存在的所有文件。

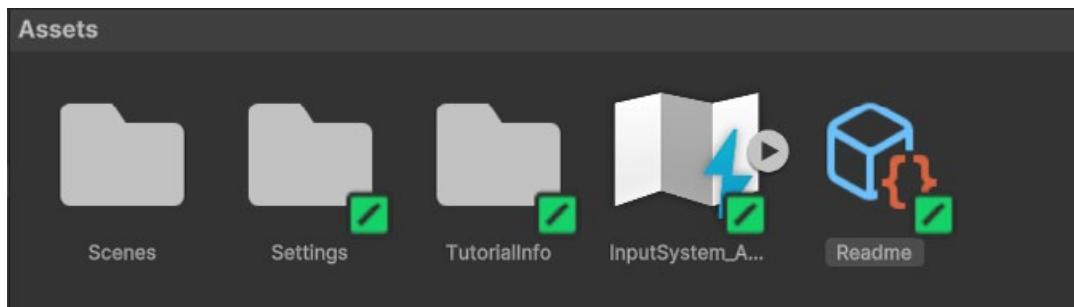
Name	Created by	Last activity	Storage used	More
Asset Manager	P	August 20, 2024	18.97 MB	...
Profile Analyzer project	P	July 30, 2024	1.47 GB	...

Unity Cloud 控制面板上的 File Explorer 显示存储库中文件和文件夹的结构。



UVCS 会忽略 Library、Logs 和 User Settings 文件夹，因为这些文件夹是在计算机上自动生成的，会占用大量空间。

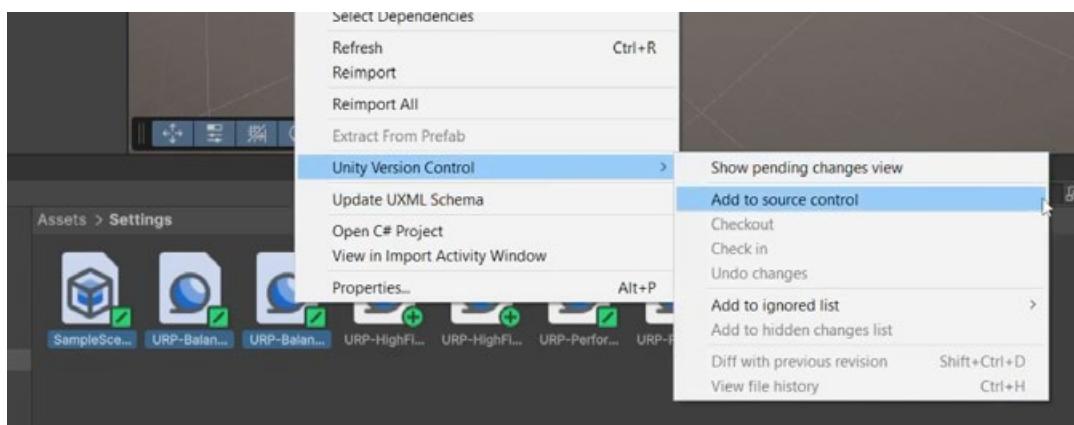
UVCS 忽略的其他文件夹将在 Project 窗口中的文件夹或文件下方有一个带有正斜杠图标的绿色框。



未在云中备份的被忽略的文件将显示一个包含正斜杠的绿色框。

如果项目使用 URP 或 HDRP，请务必将 Settings 文件夹包含在 Version Control 窗口的 Pending Changes 部分中。如果 Settings 文件夹的绿色框中有正斜杠，请选择该文件夹并右键单击。选择 Version Control > Add to source control。

正斜杠将变为绿色圆圈中的加号图标，这些文件现在将被添加到待处理更改列表中，准备上传到云端。

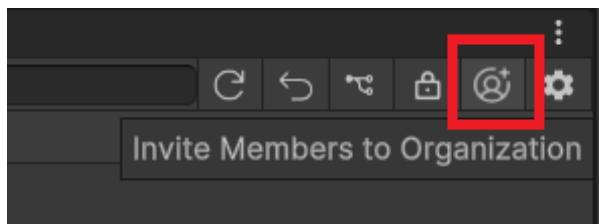


将设置文件包含在待处理更改视图中



邀请其他团队成员

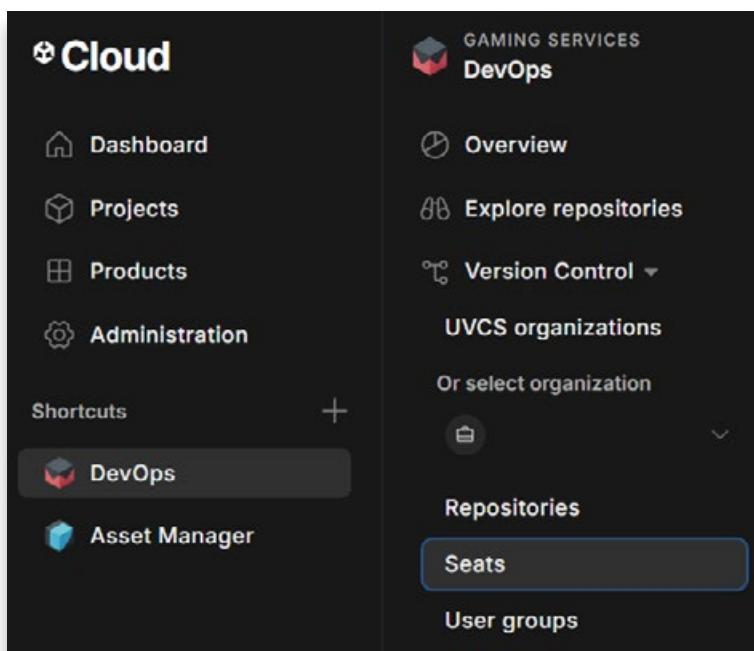
在 Version Control (版本控制) 窗口中，窗口右侧有一个按钮，可用于邀请团队成员。



“版本控制” (Version Control) 窗口中的“邀请成员加入组织” (Invite Members to Organization) 选项

这将带您进入 Unity Cloud 登录页面。使用您的 Unity ID 和密码登录。

DevOps 将从左侧的快捷方式打开，然后席位部分也将打开，允许您向当前项目添加额外的席位。在撰写本文时，您不能在免费套餐中添加超过三个席位。您可以升级到 Cloud Pro 以购买更多席位。有关更多信息，请参阅 Unity Cloud 定价计划页面，或者登录 Unity Cloud 控制面板并转到 DevOps 控制面板的“关于”页面。

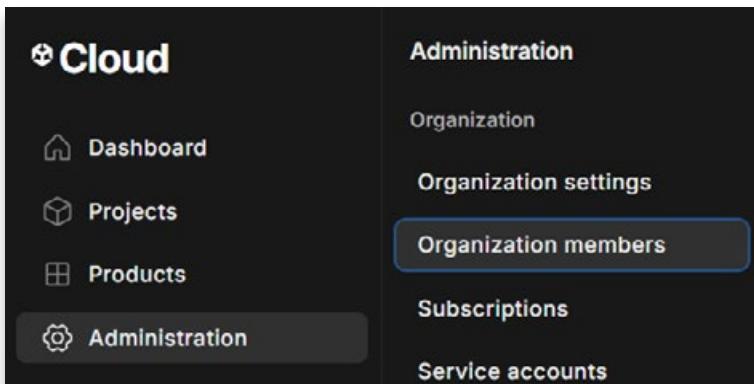


从 Unity Cloud 上的 DevOps 部分添加额外的席位。

在项目中为新成员分配席位。还应邀请他们成为组织成员。这使他们能够完全访问云控制面板上的所有数据，包括使用情况统计信息。



转到 Administration>Organization members（组织成员），然后单击 invite organization members（邀请组织成员）按钮以向组织添加额外的人员。



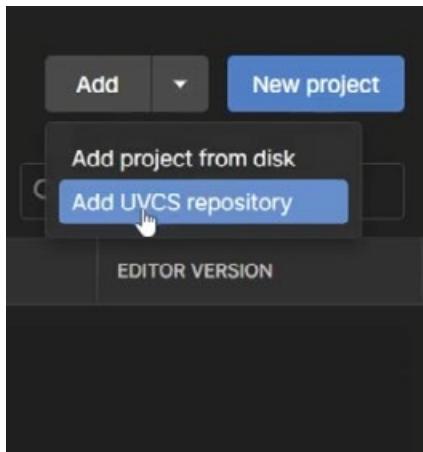
从 Administration 选项添加额外的组织成员。

为每个成员分配适当的角色。从 guest、user 或 admin 中进行选择。用户和管理员成员可以查看所有项目，您可以在项目成员部分中分配角色，以授予他们对项目数据的适当级别的访问权限。

这允许他们访问项目并根据其角色为项目做出贡献。具有管理员身份的用户将拥有对云上可用选项的完全访问权限。

系统会向您邀请的其他成员发送一封电子邮件。

然后，受邀成员应打开 Unity Hub，然后单击 Add 按钮旁边的下拉菜单，然后选择 Add UVCS repository：



单击 Add UVCS repository 以在 Unity Cloud 上打开一个项目。

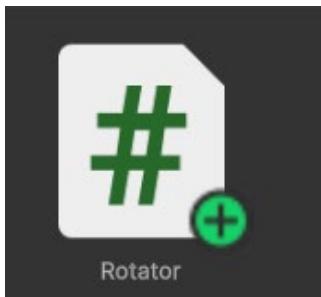
注意：所有团队成员都必须使用相同版本的 Unity，否则在版本之间转换项目时可能会导致问题。

然后，这会将项目从云存储库下载到用户的计算机上。



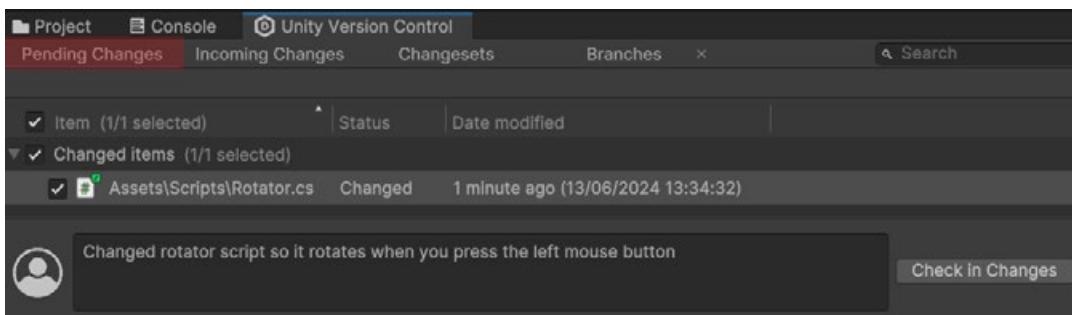
签到更改

当您对项目进行更改时，您将在更改的文件上看到一个绿色的 **+** 图标。这将显示要检入服务器以进行更新的文件。



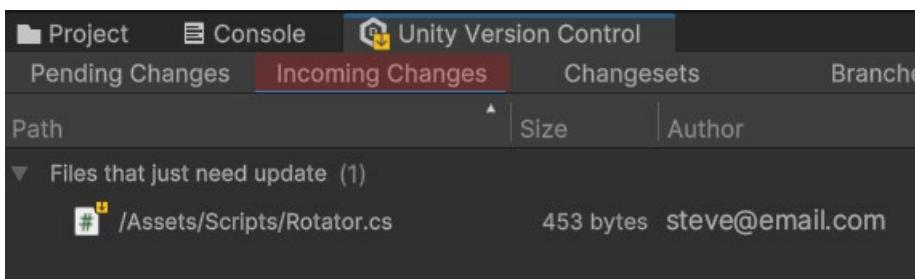
带有加号图标的绿色圆圈表示文件已更改，需要签入。

经常更新您的项目，以便更轻松地回滚。更新的文件显示在 Version Control 窗口的 Pending Changes 部分中。添加您所做的更改的简短描述，然后单击 Check in Changes 按钮。



Pending Changes 部分允许您签入对项目所做的更改。

正在处理同一分支的其他用户将在 Version Control 窗口的 Incoming Changes 选项卡下看到一个黄色的下载图标。这将显示其他用户所做的更改。

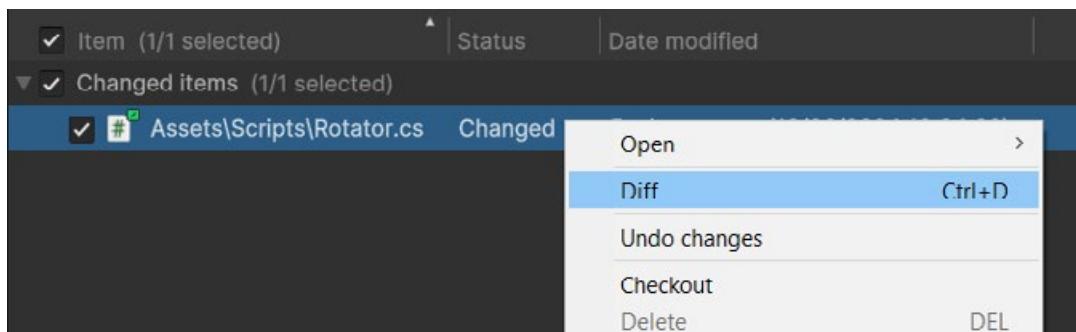


其他用户可以看到您签入的更改，并可以更新其工作区以合并这些更改。

他们可以单击 Update workspace (更新工作区) 按钮来更新其工作区并合并您的更改。



或者，如果它是已修改的现有文件，他们可以通过右键单击已更改的文件并选择 Diff. 来选择查看之前和之后的差异。



检查更改的文件与原始文件之间的差异。

在 Diff 查看器中，他们现在可以在左侧看到原始版本，在右侧看到传入的更改。这对 C# 脚本非常有用。Unity 的代码感知合并技术 Semantic Merge 会跟踪移动的代码，以帮助您仅关注相关更改。

```
cs:30
```

```
11 [Tooltip("Prefab for the primary objectives")]
12 public PoolObjectDef secondaryObjectivePrefab;
13 
14 Dictionary<Objective, ObjectiveToast> m_ObjectivesDictionary;
15 
16 public Text carPosition;
17 private int carPos = 1;
18 
19 void Awake()
20 {
21     m_ObjectivesDictionary = new Dictionary<Objective, ObjectiveToast>();
22     carPosition.text = carPos.ToString();
23 }
24 
25 public void RegisterObjective(Objective objective)
26 {
```

```
cs:28
```

```
11 [Tooltip("Prefab for the primary objectives")]
12 public PoolObjectDef secondaryObjectivePrefab;
13 
14 Dictionary<Objective, ObjectiveToast> m_ObjectivesDictionary;
15 
16 public Text carPosition;
17 public Text nitroPower;
18 
19 void Awake()
20 {
21     m_ObjectivesDictionary = new Dictionary<Objective, ObjectiveToast>();
22 }
23 
24 public void RegisterObjective(Objective objective)
25 {
26     objective.onUpdateObjective += OnUpdateObjective;
```

Diff 查看器在顶部以橙色显示原始文件，在下面以

蓝。

所有变更集都可以在 Changesets (变更集) 部分的列表中看到，显示谁在什么日期和时间进行了更改。更改的描述和已更新的文件包含在 Changeset 列表中。



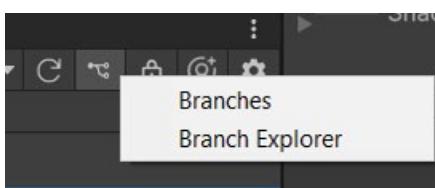
Changes	Incoming Changes	Changesets	Branches
			x
Creation date		Created by	Comment
13/06/2024 12:19:23		steve@email.com	Created a rotator script for the airplane propellor
11/06/2024 13:35:19		pete@email.com	Created a scene with spline extrusion
11/06/2024 13:07:19		pete@email.com	Added WW2 Thunderbolt asset to a new scene
10/06/2024 20:40:52		steve@email.com	Added a new spline
10/06/2024 20:35:17		steve@email.com	Added a red capsule
10/06/2024 20:34:02		steve@email.com	Setup second workspace
06/06/2024 19:33:23		pete@email.com	Root dir

对项目所做的所有变更集的显示;您可以回滚到任何变更集

UVCS 桌面客户端

Unity Editor 中的 Version Control 窗口提供了从云端推送和拉取数据所需的标准功能。桌面客户端提供了其他功能。

要启动桌面客户端，请在展开模式下工作时单击 Branch Explorer。如果您尚未安装客户端，这将提示您安装。



单击 Branch Explorer 按钮以启动 UVCS 桌面应用程序。



版本控制应用程序，包括 Gluon 和 Unity DevOps 版本控制，后者是 UVCS 桌面应用程序

Gluon 是更适合初学者的软件，而 UVCS 是功能齐全的选项。默认情况下，单击 Branch Explorer 将打开 UVCS 软件。您可以通过从计算机上安装的软件中打开 Gluon 和 UVCS 来在 Gluon 和 UVCS 之间切换。Unity 将自动检测您正在使用的版本并切换到使用该系统。



Workspace: Kart Racer

Kart Racer 33@Kart Racer
Added a black and white variation of Intro art 2

Explore workspace Checkin changes Incoming Changes Changesets

Configure Update workspace

Item	Status
e:\Udemy Unity\Version control	Controlled
Assets	Controlled
Packages	Controlled
ProjectSettings	Controlled
.vs	Ignored
Library	Ignored
Logs	Ignored
obj	Ignored
Temp	Ignored
UserSettings	Ignored
.vsconfig	Private
Assembly-CSharp.csproj	Ignored
ignore.conf	Ignored
Kart Racer.sln	Ignored
KartGame.AI.csproj	Ignored
KartGame.AI.Editor.csproj	Ignored
KartGame.csproj	Ignored
KartGame.Editor.csproj	Ignored
Unity.Karting.Tutorials.csproj	Ignored
Unity.Microgame.Tutorials.csproj	Ignored

Gluon 是一个对艺术家友好的版本控制应用程序。

Workspace: Kart Racer

Kart Racer 26@Kart Racer
Created the main race track scene

Filters: Date: Last month Branch: User: Other

24/06/2024 09/07/2024

/main

The screenshot shows the UVCS interface with a timeline of commits on the /main branch. There are seven green circular nodes connected by arrows, representing a sequence of changes. The first node has a small icon next to it, likely indicating it's the current active commit or a specific point of interest.

UVCS 是完整的软件。



使用 Gluon

如介绍 UVCS 的部分所述，Gluon 是一个专为艺术家设计的精简客户端。它允许您只选择要处理的文件并从服务器中签出它们，从而锁定它们不被其他任何人修改。完成工作后，将文件签回。Gluon GUI 删除了更复杂的概念，这些概念更适合程序员而不是技术水平较低的用户。

从已安装的 Unity DevOps 版本控制应用程序中打开 Gluon。



可以从已安装的 Unity DevOps 软件中找到 Gluon 桌面应用程序。

它将提示您从 full 模式更改为 partial 模式，即 Gluon 运行的模式。

The workspace is in full mode. Run an update (get latest) to switch to partial mode. This GUI does not work well in full mode

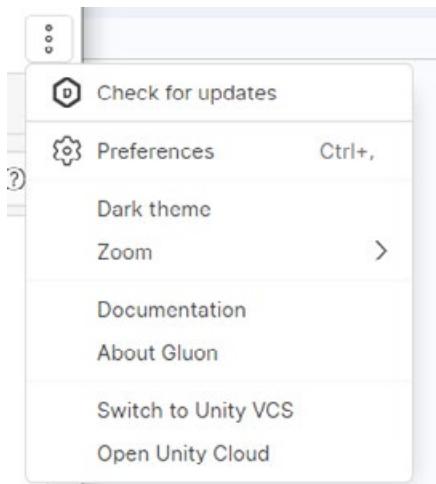
此警告允许您在 Gluon 中工作时通过单击运行 update. 更改为部分模式

在 Unity 中，Configure Gluon 按钮取代了 branches 图标。这将在您每次单击 Gluon 应用程序时打开它。



团队成员 sing Gluon 现在将能够直接从版本控制风中打开 Gluon 应用程序 now 在 Unity 中。

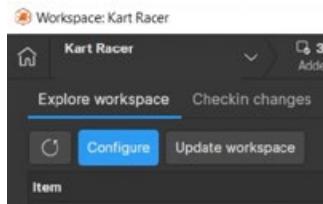
您可以通过单击 Gluon 应用程序右上角的选项按钮来启用深色主题。



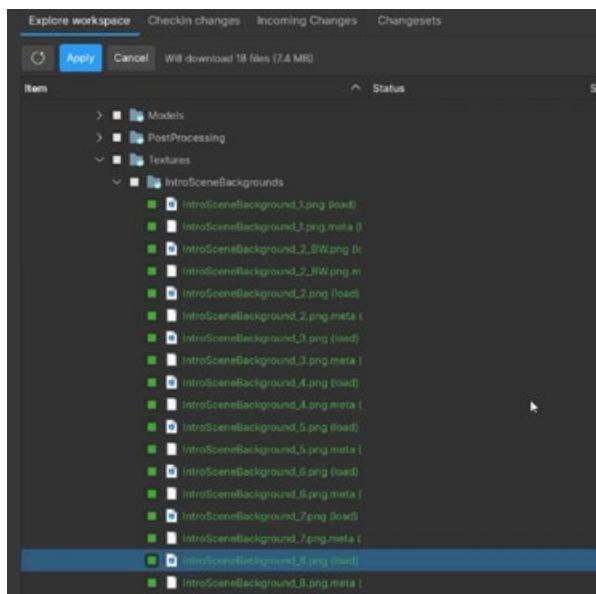
在 Gluon 中，您可以使用右上角的选项下拉菜单来启用深色主题。

Gluon 仅使用 main 分支，提供对艺术家友好的可视化工作区。您可以使用 Gluon 轻松访问“探索”工作区功能，以根据需要上传和下载文件。

在 Explore workspace（浏览工作区）部分中，单击 Configure（配置）按钮以更改存储库中的文件。



单击 Configure 按钮以更改存储库中的文件。



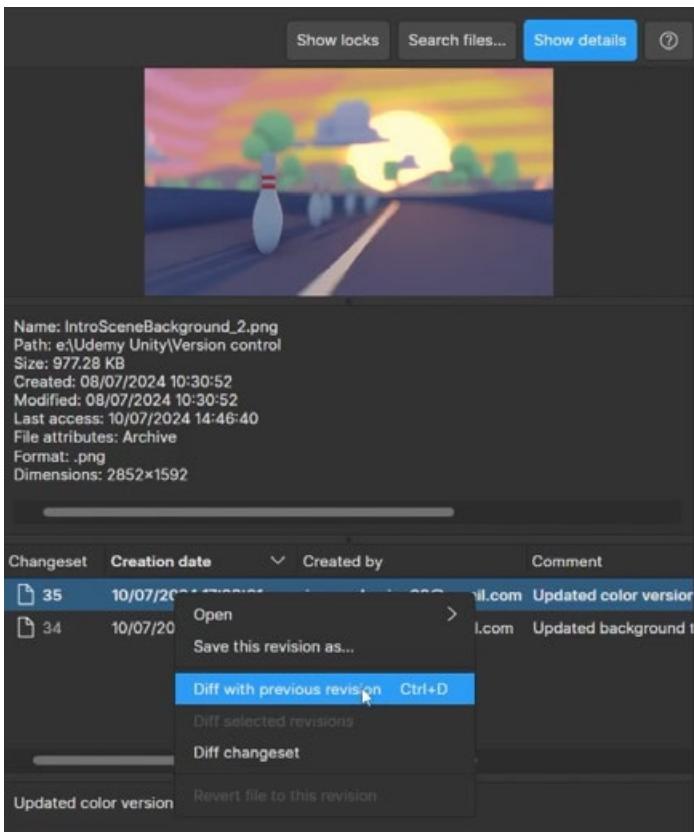
选择您需要下载的文件，然后单击 Apply in Gluon。

单击文件旁边的复选框，以下载或从 Explore 工作区视图中删除。将下载到 Unity Editor 的文件将变为绿色，要从 Unity 中删除的文件将变为红色。完成后单击 Apply，然后单击 Update workspace，以便文件在您的 Unity 版本中可用。

即使你删除了 Gluon 中的文件，它们也只与当前变更集和你创建的任何未来变更集相关。这些文件仍将位于云中，因为它们将被以前的变更集引用。

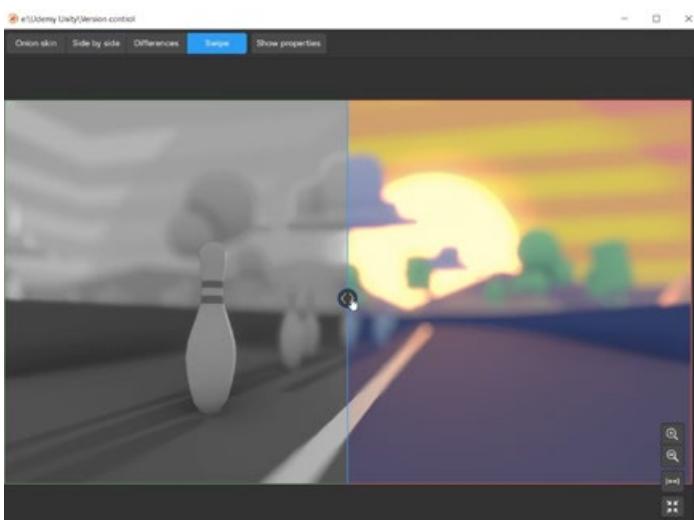


您可以通过右键单击 Gluon 应用程序右侧选项部分中的修订版，然后选择 Diff with previous revision ..，来查看具有大量修订版的纹理的视觉变化



通过选择 Diff with previous revision. 来查看修订之间的视觉更改

这将显示上一个修订版和当前修订版之间的差异。您有几个选项可以查看此版本，包括滑动，它允许您沿图像拖动滑动块以查看之前和之后。



差异查看器中的滑动选项



UVCS 桌面应用程序

UVCS 桌面应用程序是完整、完整的版本控制软件，适用于希望访问分支的团队成员。您可以从已安装的 Unity DevOps 版本控制应用程序中打开它。

如果您以前使用过 Gluon，它会提示您从部分模式更改为完整模式，即 UVCS 运行的模式。



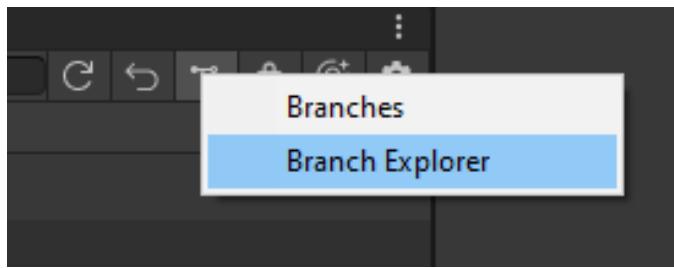
将 UVCS 桌面版本控制软件从部分模式更改为完整模式。

分支

分支允许不同的团队成员彼此独立地处理同一个项目。例如，团队成员可能在单独的分支上创建新功能，而其他团队成员则在主分支上工作。功能可以是创建成就代码、为不同类型的拾取物添加机制等。

当他们完成任务时，他们可以将他们的分支合并回 main 分支。然后，所有参与该项目的团队成员将能够更新他们的工作区以包含已完成的所有工作。

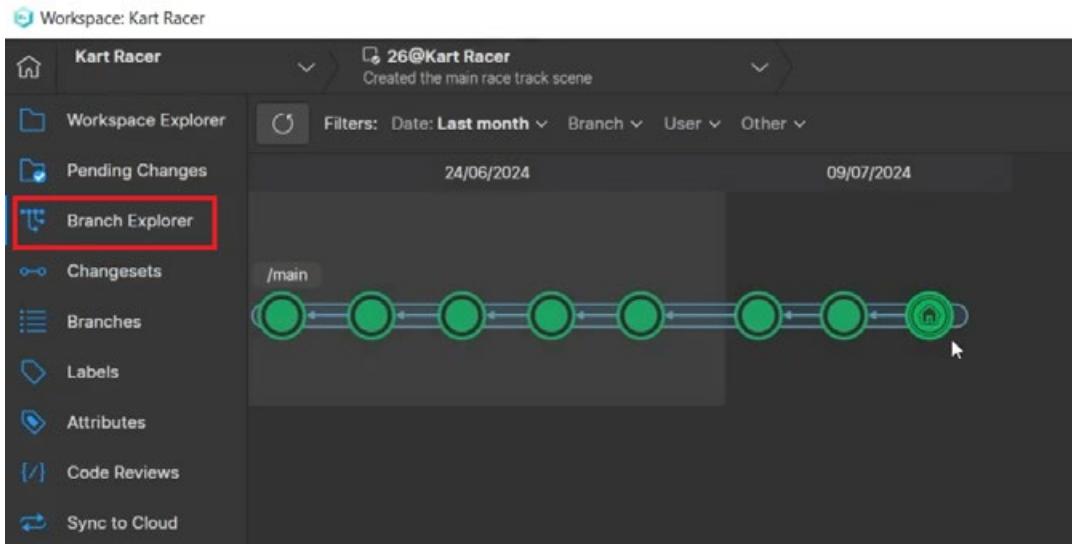
在 Unity Editor 的 Version Control 窗口中选择 Branch Explorer 时，它会打开 UVCS 应用程序并转到 Branches 部分。



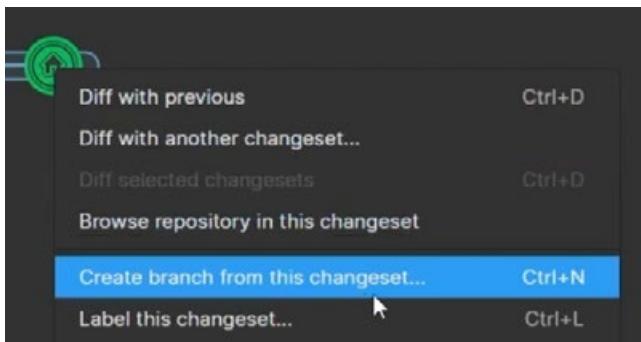
您可以打开 UVCS 桌面应用程序的 Branch Explorer 部分。



每个圆圈代表一个变更集。您当前的变更集有一个 house 图标。



右键单击变更集，然后选择 Create branch from this changeset.

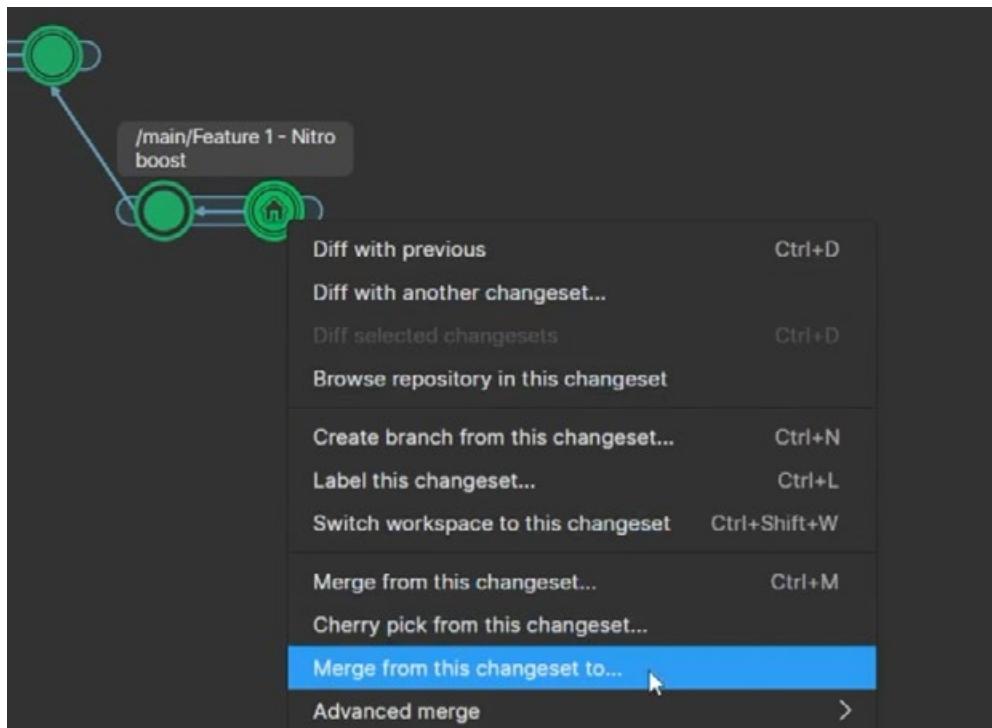


您现在将处理一个单独的分支。团队成员在 main 分支或其他功能分支上所做的任何更改都不会发送给您。这降低了其他人的更改与您的进度冲突的风险。您可以在合并后更新其更改。

分支可以有多个由圆圈标记的变更集，就像主分支一样。

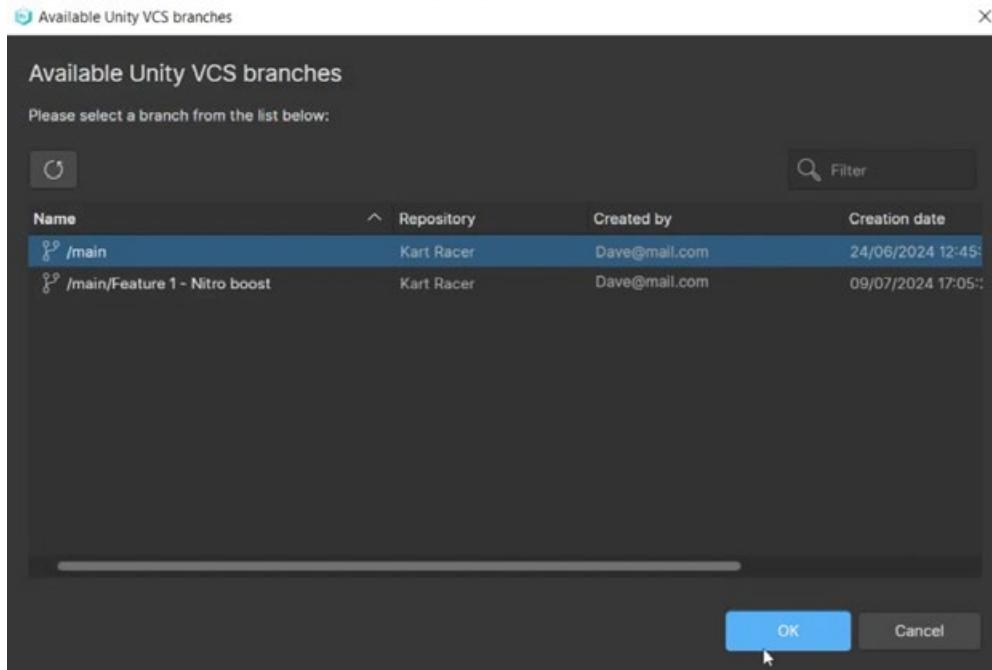


完成分支上的工作后，可以通过右键单击并选择 Merge from this changeset to....



将一个分支合并回另一个分支，例如 Main.

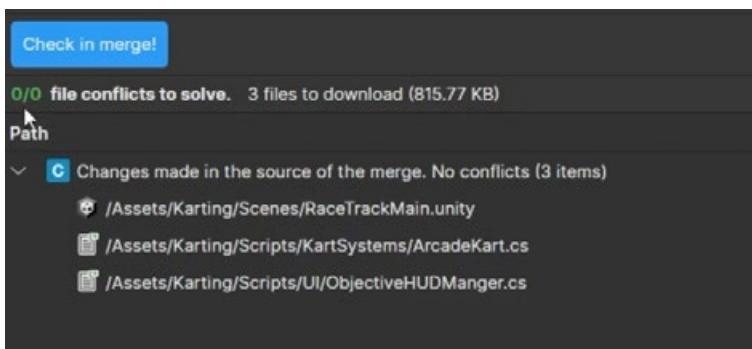
它将显示可用的分支。选择要合并到的分支。



可合并到的分支



这将打开 merge 部分并显示是否存在任何冲突。如果没有冲突，请单击 Check in merge 按钮。

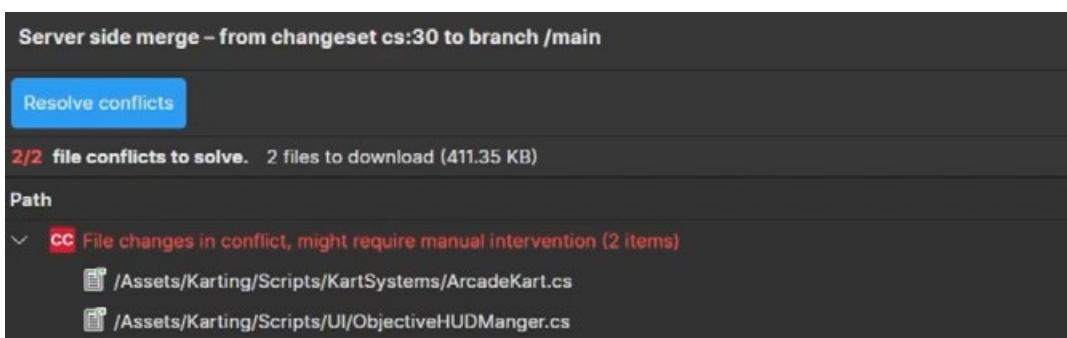


UVCS 会检查分支上的所有文件是否与 main 分支发生冲突。

如果 main 分支上的某人对您拥有的一个或多个文件进行了更改，则可能会在尝试合并时导致冲突。

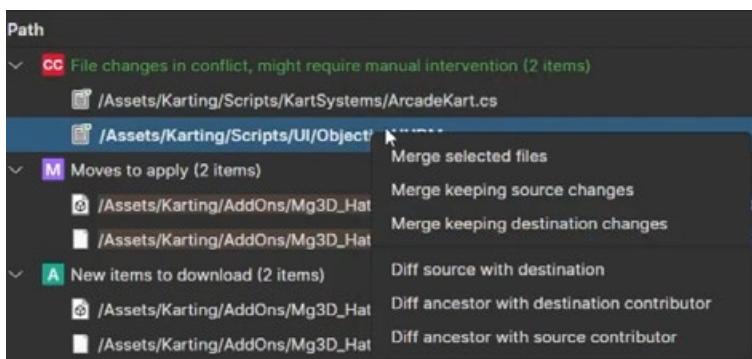
处理冲突

脚本或场景文件可能由不同分支上的两个或多个人更改。尝试将它们合并在一起将触发合并冲突。



已确定两个冲突，其中显示了包含冲突的文件。

右键单击存在冲突的文件以显示差异，然后决定是要保留还是删除更改。



显示可用于处理合并冲突的选项的右键单击菜单



使用脚本，您可以选择 Diff source with destination（将源与目标进行比较）。这将打开 Diff 查看器，以便你可以看到你的文件与你想要合并的分支上的文件有何不同。

The screenshot shows the UVCS Diff viewer interface. It displays two versions of a C# script side-by-side. The left pane is labeled 'cs:30' and the right pane is labeled 'cs:28'. The code is color-coded to highlight changes. A blue bar at the bottom indicates the difference: 'Diff 查看器 s 您的文件与所需分支上的代码之间的差异' (Diff viewer s The difference between your file and the code in the required branch) and '以合并。' (To merge.)

```
11 [Tooltip("Prefab for the primary objectives")]
12 public PoolObjectDef secondaryObjectivePrefab;
13
14 Dictionary<Objective, ObjectiveToast> _objectivesDictionary;
15
16 public Text carPosition;
17 private int carPos = 1;
18
19 void Awake()
20 {
21     _objectivesDictionary = new Dictionary<Objective, ObjectiveToast>();
22     carPosition.text = carPos.ToString();
23 }
24 public void RegisterObjective(Objective objective)
25 {
26     objective.onUpdateObjective += OnUpdateObjective;
```

```
11 [Tooltip("Prefab for the primary objectives")]
12 public PoolObjectDef secondaryObjectivePrefab;
13
14 Dictionary<Objective, ObjectiveToast> _objectivesDictionary;
15
16 public Text carPosition;
17 public Text nitroPower;
18
19 void Awake()
20 {
21     _objectivesDictionary = new Dictionary<Objective, ObjectiveToast>();
22 }
23
24 public void RegisterObjective(Objective objective)
25 {
26     objective.onUpdateObjective += OnUpdateObjective;
```

这可以帮助您决定是否使用您的文件覆盖另一个分支上的文件。为此，请选择 Merge keep source changes（合并保留源更改）：

Merge keeping source changes

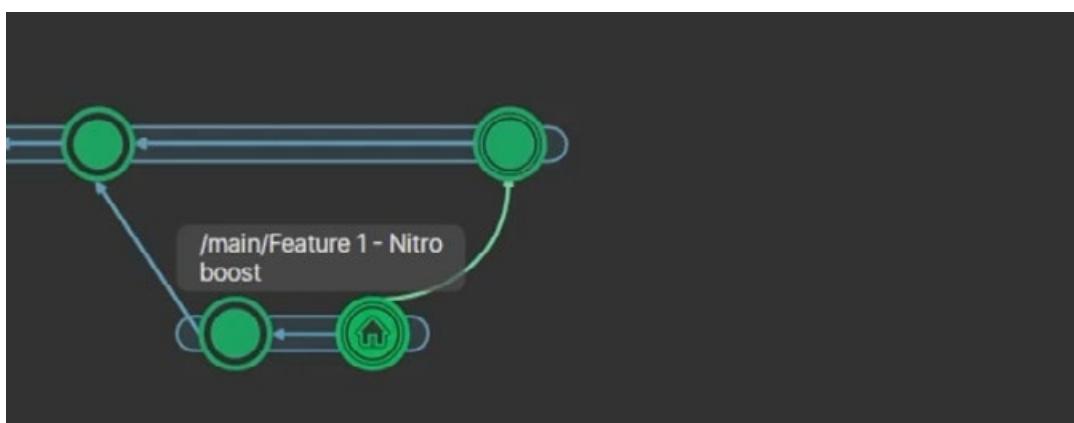
或者，您可以放弃更改并将原始文件保留在另一个分支上。为此，请选择 Merge keep destination changes（合并保留目标更改）：

Merge keeping destination changes

最后，您可以选择 Merge selected files（合并所选文件），UVCS 会将两个文件合并在一起，尝试保留两组更改，但您应该在合并后检查脚本以确保它已正确完成此作：

Merge selected files

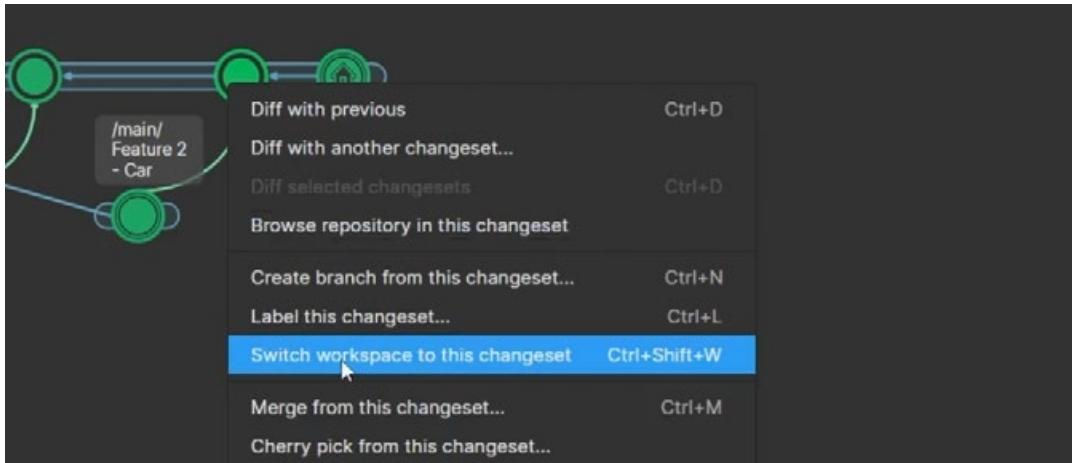
Branch Explorer 现在将显示哪些变更集继承了其他分支的合并。



Branch 合并回 Main



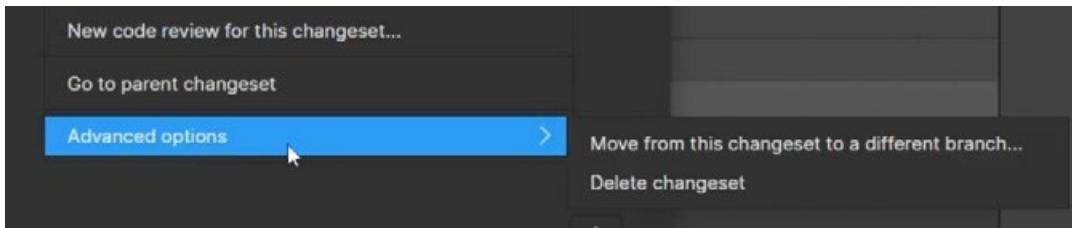
您还可以右键单击任何以前的更改集，然后选择 Switch workspace to this changeset（将工作区切换到此更改集）以恢复到该点。如果您添加的项目或脚本会破坏游戏或导致问题，则可能需要这样做。



您可以通过右键单击所选变更集来切换回任何以前的变更集。

导致严重错误或 bug 的变更集可以删除。这些变更集不应有任何从它们向前的变更集，也不应有标签或子分支。

右键单击有错误的变更集，然后转到高级选项 > 删除变更集。



从右键单击菜单中删除变更集。

合并规则

一个好主意是实施合并规则，以防止合并冲突中断团队的工作流程。这可用于团队经理希望在允许将分支合并到主分支之前批准分支的大型团队中。这可以防止将错误引入主工作流程。



登录 Unity Cloud 控制面板，转到 settings，然后 Merge rules.

The screenshot shows the Unity Version Control interface. On the left, there's a sidebar with options like Overview, Explore repositories, Version Control (with UVCS organizations), and Settings (which is currently selected). The main area is titled 'Organization settings' and has tabs for Lock Rules, Integrations, Merge Rules (which is highlighted with a blue underline), Mergebots, and Network Allow Li. Below these tabs, there's a section titled 'Merge rules' with a sub-section 'Merge rules'. It includes a note: 'Merge Rules define conditions that branches must meet in order to be merged into other branches.' There's a 'Show more' link and a blue button labeled '+ New merge rule'.

从 Unity Cloud 设置创建新的合并规则。

单击 New merge rule 按钮，然后从四个选项中选择一个。Only Allow merge if reviewed 将确保合并只能在其他团队成员或团队经理审核和批准后进行。

如果您正在处理多个存储库，您还可以将其应用于所有存储库，或者从下拉列表中指定特定存储库。

The screenshot shows the 'New Merge Rule' dialog box. At the top right are buttons for 'Enable rule' (with a checked checkbox), 'Cancel', and 'Save rule' (in a blue button). The main area is titled 'Merge condition' with the sub-instruction 'Condition will apply to the repositories you specify.' Below this are four radio button options: 'Only allow merge if reviewed' (selected, indicated by a blue outline), 'Restrict merge to branch', 'Only allow merges from parent', and 'Only allow merges from children'. Each option has a detailed description below it. At the bottom, there's a section titled 'Apply condition to these repositories:' with a dropdown menu set to 'All repositories'.

选项 Only allow merge if reviewed 要求其他团队成员在合并前查看您的分支。

指定要将合并规则应用于的分支。它需要一个源分支，即您正在处理的分支，以及一个目标分支，即您要合并到的分支。点击 Save rule 按钮，它现在将处于活动状态。您可以随时从 Unity Cloud 中禁用或删除规则。



The screenshot shows the 'Merge Rule' configuration screen. It is divided into two main sections: 'Source branches' and 'Destination branches'.
Source branches: This section is titled 'Allow merge from this branch(es)' and contains a 'Branch pattern' input field with 'Development' selected. A note says 'Use * to specify patterns. e.g. main*'. Below it is an 'Attributes' section with a '+ Add attributes' button.
Destination branches: This section is titled 'Allow merge into this branch(es)' and contains a 'Branch pattern' input field with 'main' selected. A note says 'Use * to specify patterns. e.g. main*'. Below it is an 'Attributes' section with a '+ Add attributes' button.
At the bottom right are three buttons: 'Enable rule' (checked), 'Cancel', and 'Save rule'.

包括合并规则将应用于的分支。

想要合并其分支的团队成员必须请求分支审核。

转到 Branches 部分并右键单击要合并的分支。选择选项 New code review for this branch.

The screenshot shows the 'Branch Explorer' section of the UVCS interface. A context menu is open over a branch named '/main/Development/Feature - animated cut s'. The menu items include:

- Create child branch... (Ctrl+N)
- Switch workspace to this branch (Ctrl+Shift+W)
- Merge from this branch... (Ctrl+M)
- Merge from this branch to...
- Advanced merge >
- Diff branch (Ctrl+D)
- Diff with another branch...
- View changesets in this branch
- Push/Pull >
- Rename... (F2)
- Delete

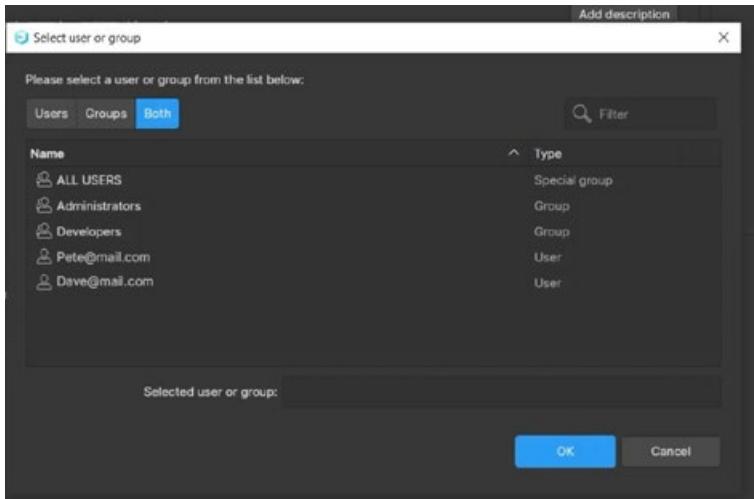
A specific item, 'New code review for this branch...', is highlighted with a blue background.

请求对要合并的分支进行代码审查。



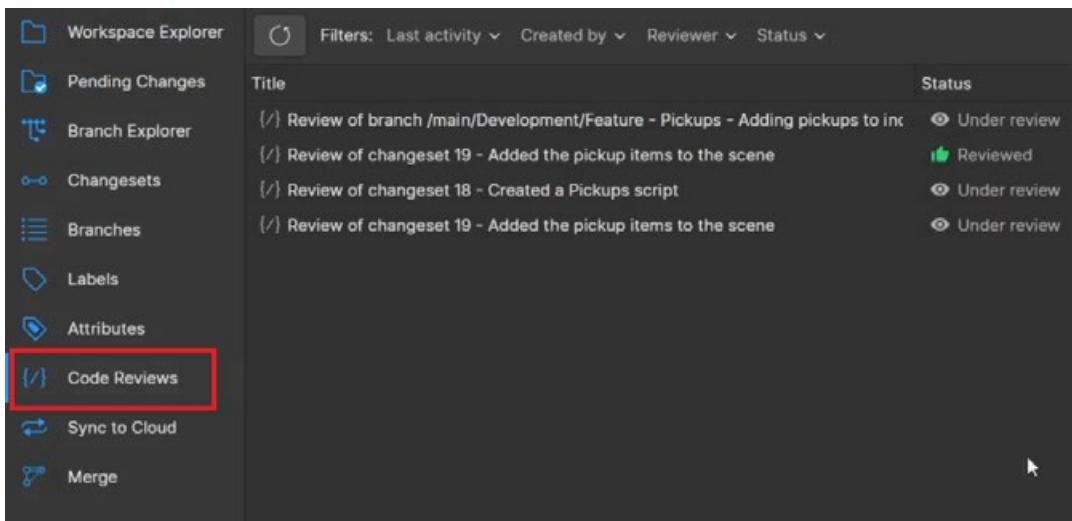
这将包含在整个分支过程中更改的所有文件。

单击对话留言并邀请其他团队成员查看您的工作。请注意，每个团队成员必须先审核分支，然后才能合并，因此请确保仅包含相关人员。然后，他们将收到一封电子邮件，以检查您的代码。



选择个人或组成员以查看您的分支。

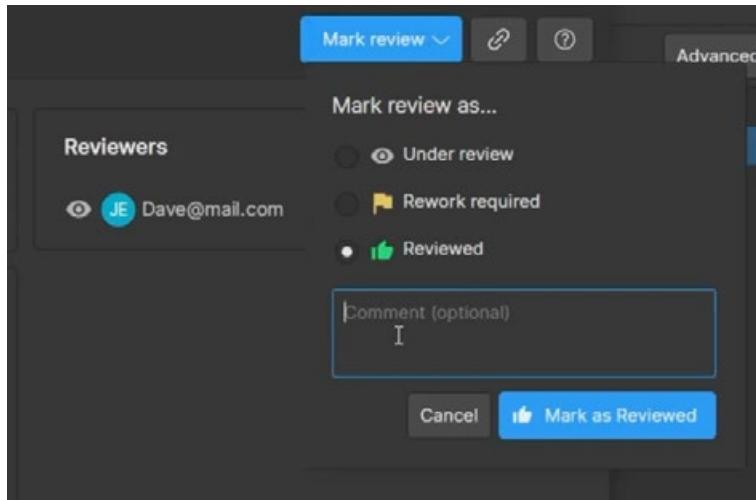
转到 Code Reviews（代码审查）以查看审查的进度。双击将其打开，查看是否为您留下了任何消息。



代码评审显示请求的评审的持续进度。



然后，受邀的团队成员可以将分支标记为 Rework required（需要返工）或 Reviewed（已审核）并留言。

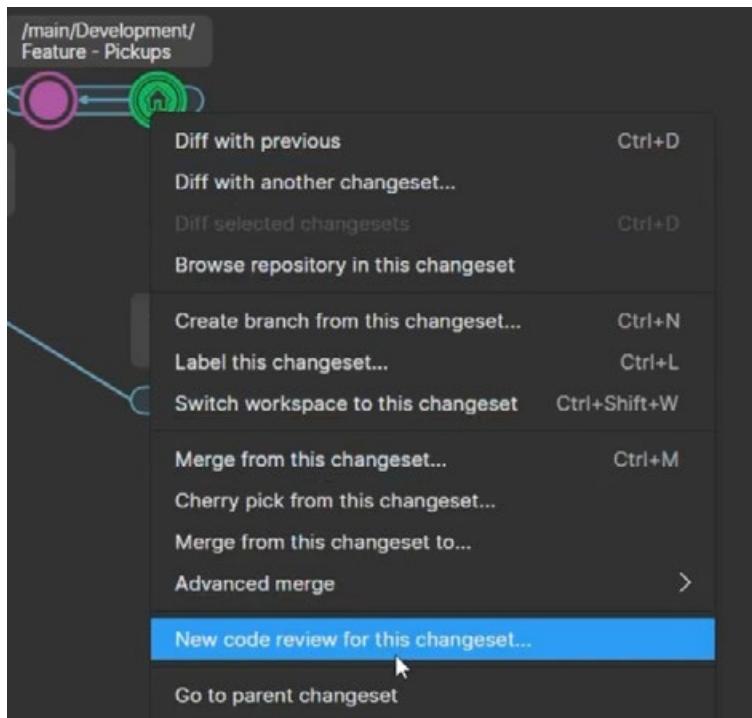


审阅者可以单击 Mark Review 按钮将其标记为 rework required 或 reviewed。

一旦 Branch 被审查，就可以合并它。这为您的项目增加了额外的安全性。

您还可以对特定变更集的代码审查执行相同的过程。当您希望其他团队成员检查您的代码中是否存在 bug 或问题，或者建议改进代码的方法时，这可能很有用。

为此，请右键单击变更集，然后选择 New code review for this changeset（此变更集的 New code review）。然后，您可以邀请团队成员检查您的文件并提供建议。

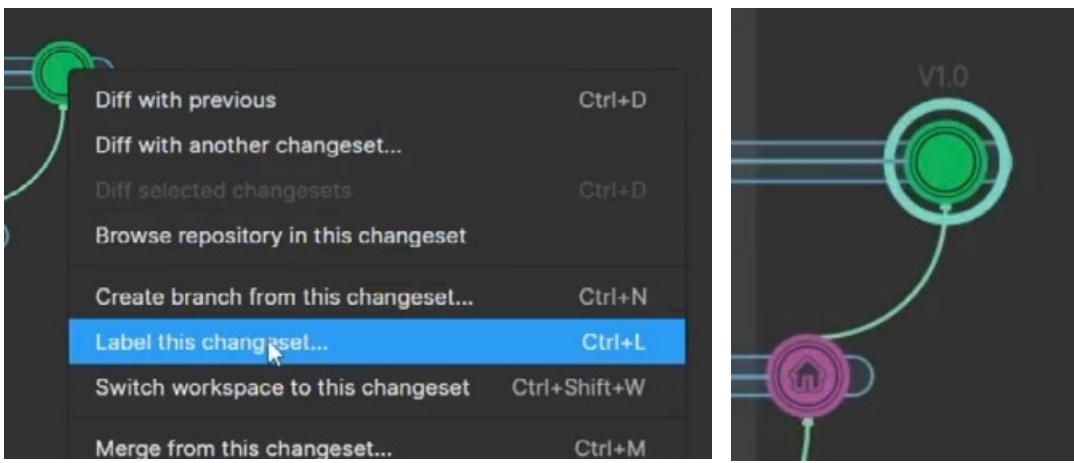


请求对变更集进行代码审查



您还可以向变更集添加标签。如果变更集是已批准的版本（例如 1.0 版），或者它是一个稳定的变更集并且可以用于分支，这可能很有用。

右键单击变更集，然后选择 Label this changeset（标记此变更集）。标签可防止删除变更集。如有必要，您可以从 labels 部分删除标签。



从右键单击菜单中标记变更集。

标签为其他用户提供重要信息
关于此 changeset.

锁定文件

在处理文件时锁定文件可以防止重大合并冲突。然后，此文件将被锁定在云中，以防止其他团队成员在锁定时对其进行更改（他们仍然可以以只读格式访问文件）。重新签入文件后，锁定将被删除，其他团队成员可以对其进行更改。

锁不会自动应用。默认情况下，当您在 Unity 中保存进度时，您正在处理的任何文件都将标有签出图标。

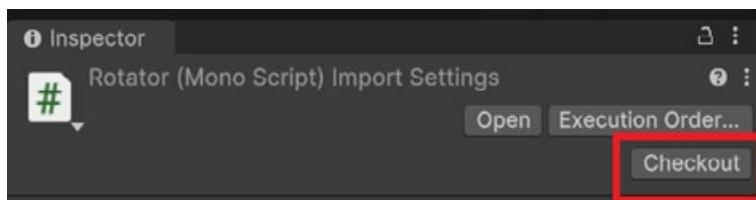


一个紫色的签出图标出现在 Unity 中的文件图标上，也出现在其他所有人的版本上，让他们知道有人正在处理该文件。

紫色框中的结帐图标显示在每个团队成员的 comp 上的文件上

子宫

您可以在开始处理文件之前手动签出文件。单击 Inspector 中的 Checkout 按钮。其他团队成员将看到您正在处理该文件，但是，这不是文件锁定，因此其他团队成员仍然可以对同一文件进行自己的更改。这不如使用文件锁定有效。



中若您进行更改时，请 eckout 一个文件以警告其他团队成员您正在

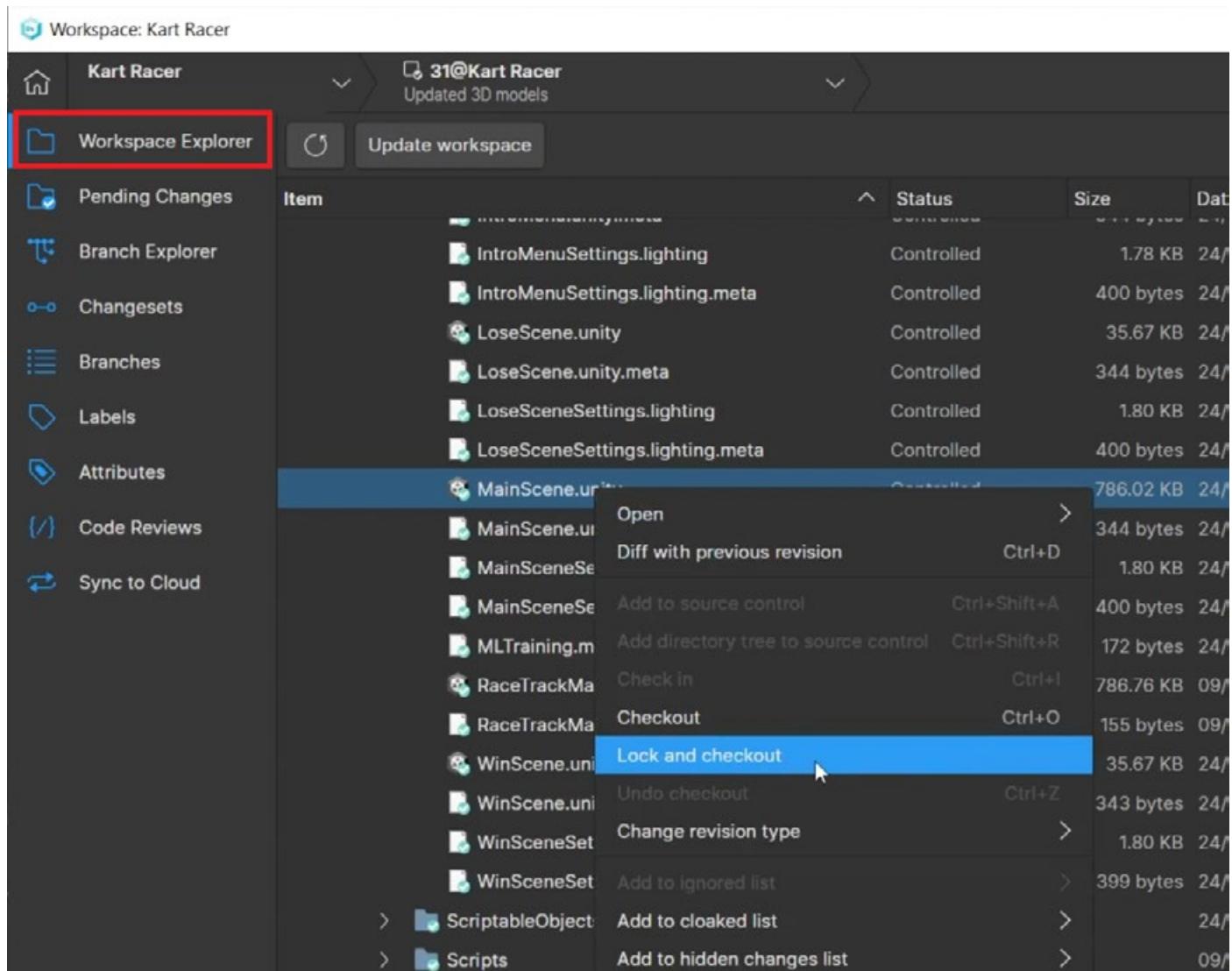
在上面嬉戏。



UVCS 使用智能锁。这可确保要更改此文件的其他团队成员将被迫使用该文件的最新版本。这很重要，因为您可能正在处理使用该文件的旧版本的分支。智能锁会检查包括主分支在内的所有分支以查找此文件的最新版本，并确保您使用的是该版本，而不是过时的旧版本。

要锁定文件，请进入 Branch Explorer。

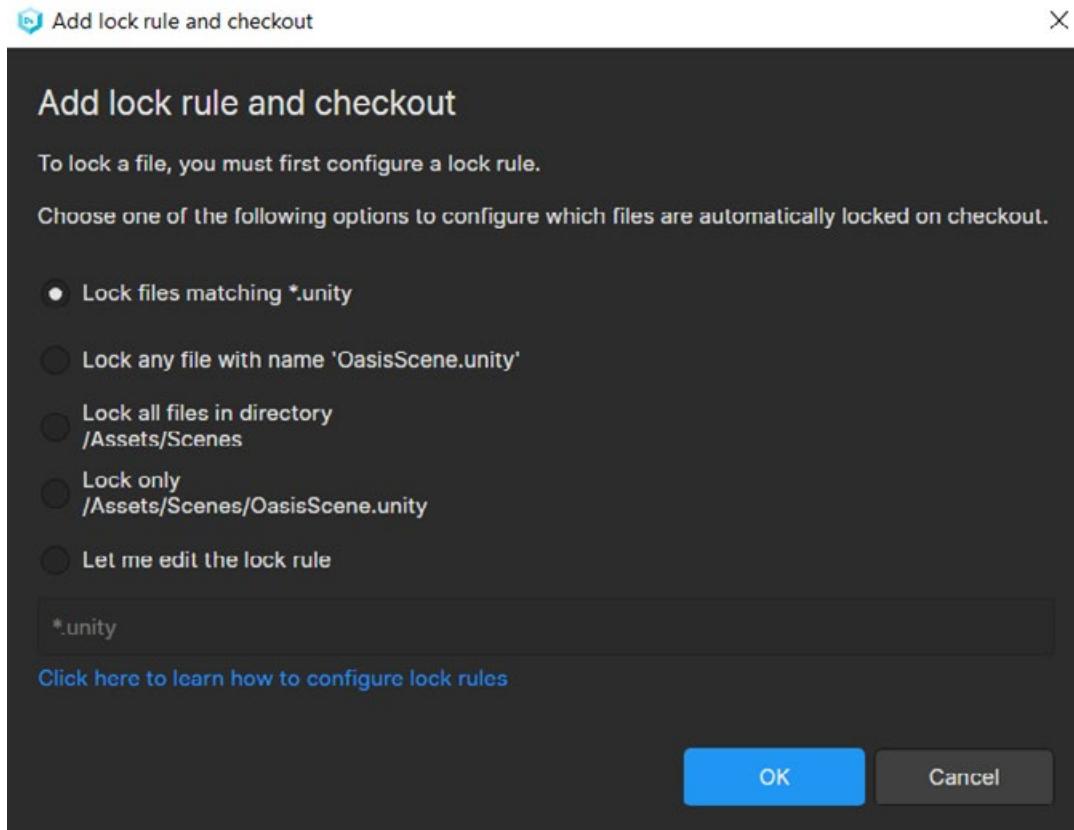
在 Workspace Explorer 视图中，找到要锁定的文件，右键单击它，然后选择 Lock and checkout。仅当没有其他团队成员签出文件时，此选项才有效。



在 Workspace Explorer 中从 UVCS 桌面应用程序锁定文件。



您可以为此类文件定义锁定规则。要仅锁定此特定文件，请选择 Lock only（仅锁定）（下面列表中的第四个选项）。



定义用于锁定此文件或文件类型的规则。



现在，该文件将在每个团队成员的 Unity 版本中显示一个紫色锁图标，让其他人知道您已锁定文件以进行编辑。

紫色锁图标向其他团队成员显示此文件已锁定。

团队成员仍可以只读方式查看文件，但在您重新签入文件之前无法进行任何更改。

完成更改后，签入文件，它会删除锁。



具有管理员身份的团队成员可以登录云并转到 Repositories（存储库），然后转到 File locks（文件锁定）以手动删除锁定。

Asset Manager

File locks

File Locks

These files were locked upon checkout to prevent merge conflict. You can [manage or configure lock rules](#) in your settings.

Search file locks

Lock branch Lock owner Date

Item	Lock status	Lock branch	Lock date	Lock owner
/Assets/Karting/Scenes	Locked	/main	6/24/24, 5:59 PM	Dave@mail.com

Release Remove

Rows per page: 25 1-1 of 1

删除云上的文件锁定

监控或删除存储库

您可以在桌面应用程序中监控存储库。单击左上角的 Home 图标，然后选择顶部图标以查看当前存储库。

在这里，您可以将整个存储库下载到您的计算机上，在桌面应用程序中打开它，重命名或删除它。存储库在被删除之前会在云上再保留 10 天，以防你改变主意。您可以通过选择 Undelete 来检索已删除的存储库。

Unity DevOps Version Control

Repositories +

Kart Racer RPG Game

Kart Racer

Download repository Show locks

Assets Packages ProjectSettings

Download this repository Open this repository Rename F2 Delete Delete

Undelete

Create new sync view (push/pull) Ctrl+C Path permissions Ctrl+P Permissions Repository server permissions

下载、删除或取消删除存储库



您可以在 DevOps 部分查看云上的任何存储库。这将显示实际使用的存储空间，该存储空间应比计算机上的项目小得多。

您还可以在此处添加或删除存储库。DevOps 部分将为您提供有关 DevOps 帐户的更多信息，包括使用情况报告和设置。

The screenshot shows the Unity Version Control interface with the path "Unity Version Control > Repositories > Asset Manager". The main title is "Asset Manager". There are buttons for "Star" and "Settings". Below the title, there are tabs: "File explorer" (which is selected), "File locks", "Code reviews", "Branches", and a "Back" button. A search bar with placeholder text "Search" and a dropdown menu showing "/main" are also present. On the left, there's a sidebar with links to "Assets", "Packages", and "ProjectSettings". The main area displays a table of storage usage:

Name	↑	Changeset
Assets		cs:9
Packages		cs:3
ProjectSettings		cs:5

通过登录 Unity Cloud DevOps 部分查看存储库的存储。

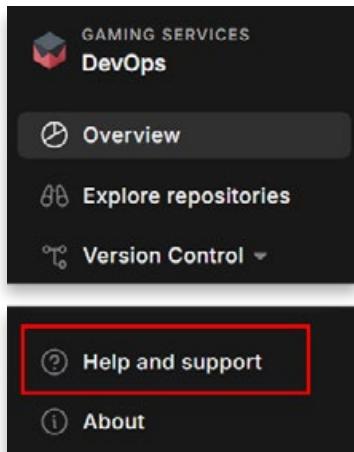
您可以通过检查 cloud. 中的“概述”部分来查看当前数据使用情况，这将显示您每月的 GB 小时数。有关更多信息，请参阅 Unity Cloud 定价计划页面，或登录 Unity Cloud 控制面板并转到 DevOps 控制面板的“关于”页面。

The screenshot shows the Unity Cloud DevOps Overview page. The left sidebar includes "Overview", "Explore repositories", "Version Control", "UVCS organizations", "Or select organization", "Repositories", "Seats", and "User groups". The main area has a "Free plan" badge and a "Upgrade plan" button. It features three cards: "Build usage" (0/200 Windows build minutes), "Storage usage for current billing cycle" (856/3720 GB-Hours), and "Build health" (2 successful, 0 failed). A "Customize" button is also present.

Cloud 上的 Overview (概述) 部分显示您的每月存储量。



Unity 支持



在 Unity Cloud 控制面板的 DevOps 部分中，单击 Help and support。

在云控制面板中，单击 Help and support 按钮以查看常见问题解答、论坛和博客文章，或生成支持票证以帮助解决您可能遇到的问题。

您还可以在 DevOps 类别的 Explore repositories 下找到示例存储库项目。您可以将这些公共存储库作为模板项目下载到您的计算机上进行处理。

The screenshot shows the Unity Cloud DevOps interface. On the left, there's a sidebar with various options like Overview, Explore repositories (which is highlighted with a red box), Version Control, and others. The main area is titled 'Explore Unity repositories' and shows a list of sample repositories under the heading 'Unity Samples' (Alpha). The repositories listed are:

Name	Stars	Last activity	Size
Megacity-Metro	46 stars	6 days ago	2.12 GB
2d-game-match3	40 stars	August 23, 2024	75.96 MB
UGS-use-case-samples	20 stars	July 31, 2024	8.81 MB
2d-game-farming	38 stars	July 19, 2024	44.8 MB

您可以下载的一些公共示例存储库

为您的实时游戏奠定基础

在 DevOps 之外，Unity 还提供了一个完整的生态系统，用于通过 Unity Gaming Services（UGS）管理实时游戏，这可以帮助您跨多个平台大规模获取和连接玩家，并管理您的后端基础设施。以下是一些服务的简要概述。

Unity 游戏服务

多人游戏

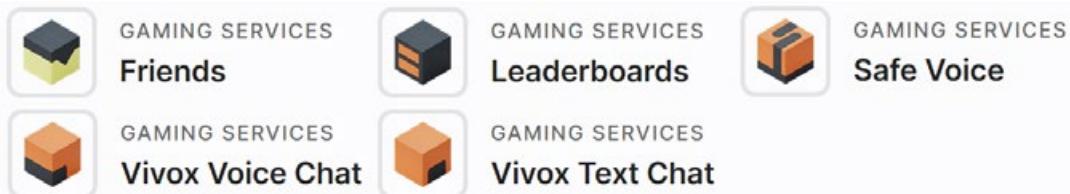


Unity 提供两个网络堆栈：用于游戏对象的 Netcode 和用于实体的 Netcode，以及来自 Unity Gaming Services 的多人游戏工具，例如 Game Server Hosting 和 Vivox Voice Chat。您可以在 Unity 的生态系统中构建，也可以根据需要混合和匹配您喜欢的工具和服务。

[了解有关 Unity Multiplayer 的更多信息](#)



社区



访问一套适用于各种规模游戏的社区游戏服务，从两人项目到数百万并发用户，以及与引擎无关的语言和文本聊天。在游戏中添加好友和排行榜等社交元素，以提高玩家的参与度和留存率。

[了解有关 Unity 游戏社区解决方案的更多信息](#)

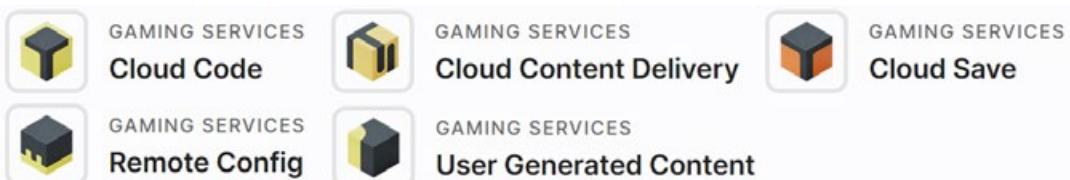
帐户



Unity Authentication 和 Unity Player Accounts 通过提供适合您的游戏和目标平台的混合搭配功能，让您可以灵活地处理玩家的身份。

[了解有关 Unity 身份验证和玩家帐户的更多信息](#)

内容管理



Cloud Save 可以高效地创造更好的玩家体验，玩家可以将游戏数据存储到云中。借助 Cloud Content Delivery，您可以通过云构建和发布具有强大资产管理和内容交付功能的游戏更新，并且每月前 50 GB 的带宽是免费的。

Remote Config 服务通过使用 Game Overrides（游戏覆盖）可以高效地运行针对特定玩家细分的 A/B 实验。在进行任何代码更改或应用更新之前，您可以更改所有玩家或部分玩家的游戏难度、广告频率或常规属性。

[Learn more about Unity Content Management](#)



崩溃报告



Cloud Diagnostics 服务有助于提高稳定性并减少客户流失。它提供实时崩溃数据，以便您的团队可以更快地修复错误并帮助防止玩家流失。Unity 提供崩溃和错误报告解决方案，这些解决方案可以免费开始使用，并且可以随着项目的增长而扩展。

[了解有关 Unity Cloud Diagnostics 的更多信息](#)

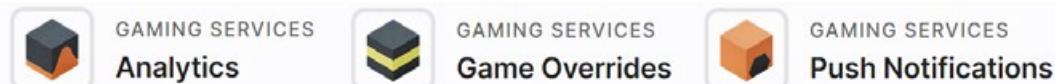
游戏经济



Game Economy 服务为您提供构建自定义游戏内经济的功能，并为玩家提供无缝购买、货币转换、库存管理等功能。将功能齐全的经济系统集成到游戏中后，您可以通过简化的仪表板有效地管理它。

[了解有关 Unity 的 Economy 服务的更多信息](#)

参与度和分析



了解您的游戏并吸引您的玩家。这些服务可以帮助您有效地优化游戏体验、留住玩家并发展您的游戏。

[Learn more about Unity Analytics](#)



Unity 成长

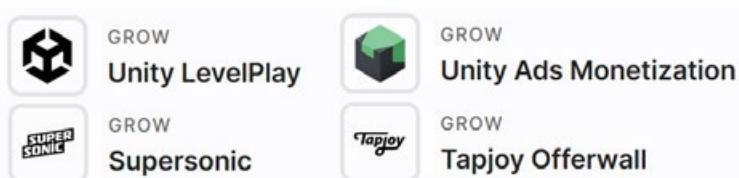
用户获取



Unity 的用户获取服务套件可以帮助您的游戏获得合适的移动用户。利用 Unity Ads 的广泛全球供应来优化您的广告活动目标，以实现广告支出回报率（ROAS）、留存率或规模。

[了解 Unity 用户获取](#)

货币



利用市场上最先进的技术和强大的工具集，有效地实现游戏变现，充分发挥应用的收入潜力。

[了解 Unity 变现](#)

结论

希望本指南能帮助您在 Unity 中作为团队的一员更自如地使用版本控制。即使您正在处理一个单独的项目，组织项目和使用版本控制的原则也非常有用。

最大的收获是清晰的团队沟通以及就工具、流程、结构和代码风格达成一致的重要性。作为一个团队，您需要就您的指导方针达成一致：您应该如何构建您的项目、使用哪个版本控制系统以及您在该系统中的工作流程是什么样子的。然后，当您开始集成其他工具（如 JIRA、GitLab、构建工具或自动化测试）时，您已经完成的构建项目和工作流程的工作将真正发挥作用。

最后，请查看以下资源，以查找有关书中讨论的各种版本控制系统的大量信息，以及有关设置 Unity 项目以取得成功的更多提示。

其他资源

— 选择版本控制系统时要考虑的八个因素 — 版本控制简介，Unite Now 2020 — Git Apprentice，作者：Chris Belanger 和 Bhagat Singh — 使用 Unity 的 Plastic SCM 对游戏进行版本控制 — Plastic SCM 产品文档 — Plastic SCM 中的 Mergebot — 具有版本控制的 Unity 开放项目 — KO_OP 如何使用 Plastic SCM 加速生产 — 扰乱发布时间表的隐藏生产力成本

Perforce 设置

— 如何配置 Helix Core 和游戏引擎 — Helix Core 文档

