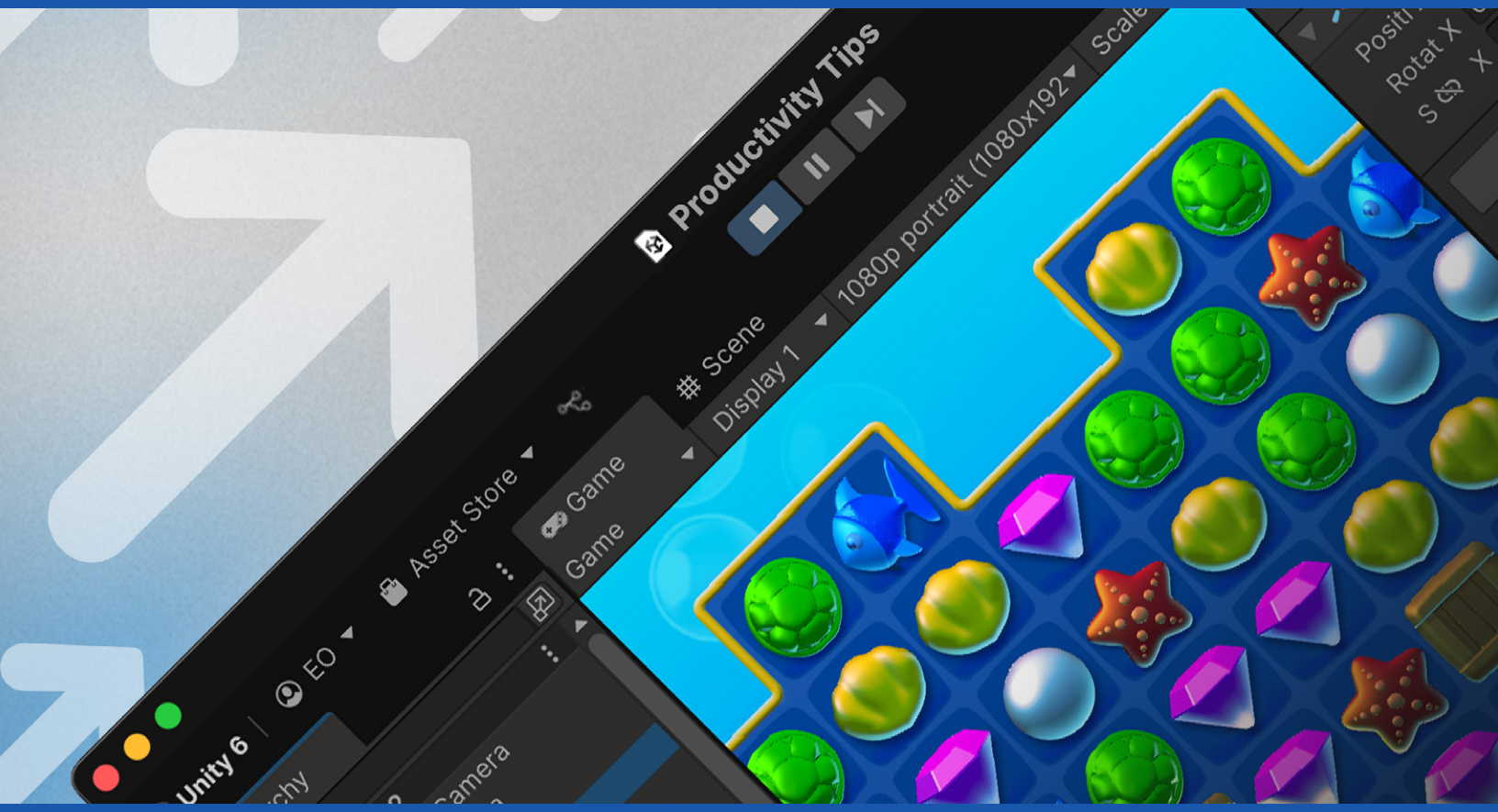
→ ЭЛЕКТРОННАЯ КНИГА

Советы по повышению продуктивности в Unity 6



Unity®

Содержание

Введение

В этом руководстве собрано множество советов, которые помогут вам быстрее работать с инструментами Unity, будь вы художник или программист. Здесь рассматриваются новые возможности Unity 6, а также проверенные способы экономии времени и рабочие процессы, которые используются в Unity уже много лет.

Когда вы работаете в Unity каждый день, самостоятельно или в команде, каждая секунда и каждый щелчок мыши имеют значение. Мы хотим помочь вам не тратить время впустую и работать продуктивнее. Независимо от вашего опыта разработки в Unity это руководство поможет ускорить рабочие процессы на каждом этапе создания игры.

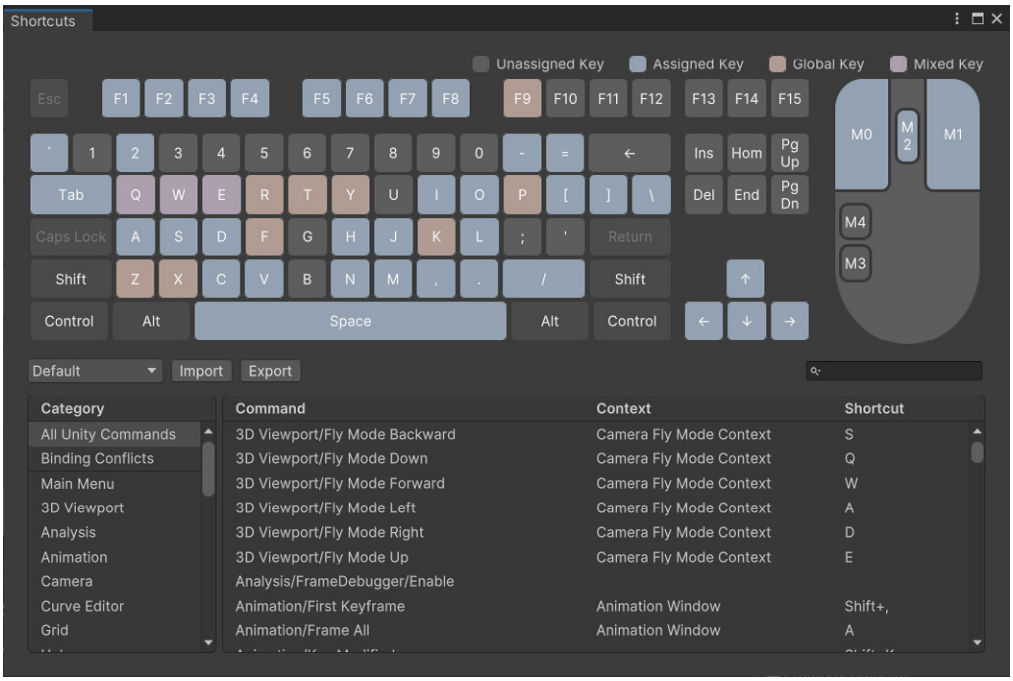
Перейдите на веб-страницу Unity 6 или к разделам «Новое в Unity 6.0» и «Новое в Unity 6.1» документации Unity, чтобы подробно ознакомиться с новыми функциями и возможностями Unity 6.

Надеемся, это руководство будет вам полезно.

Рабочие процессы в редакторе

Легко настройте удобные сочетания клавиш

Shortcuts Manager - интерактивный визуальный интерфейс для просмотра, настройки и управления горячими клавишами редактора. Он позволяет назначать сочетания клавиш для различных контекстов, например Global, окна Scene или отдельных инструментов редактора, а также просматривать текущие назначения для часто используемых инструментов. Кроме того, профили сочетаний можно импортировать и экспортировать, чтобы использовать их в команде или на разных устройствах.



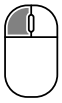
Окно Shortcuts Manager

Откройте Shortcuts Manager из главного меню Unity:

- В Windows и Linux выберите Edit > Shortcuts. .
- В macOS выберите Unity > Shortcuts. .

Основные сочетания клавиш

Ниже приведены некоторые стандартные сочетания клавиш:

Действие	Windows	Mac
Показать выбранное	F	
Дублировать элементы	Ctrl + D	⌘ + D
Удалить GameObject	Shift + Del	⌘ + Del
Вид/перенос/поворот/рамка/трансф.	Q W E R T	
Переключить режим опорной точки	Z	
Переключить ориентацию опорной точки	X	
Привязка к вершинам	V	
Привязка	Ctrl + 	⌘ + 
Изолировать выбранные объекты в окне Scene	Shift + H	
Развернуть или восстановить окно	Shift + Space	
Редактировать префаб в контексте	P	

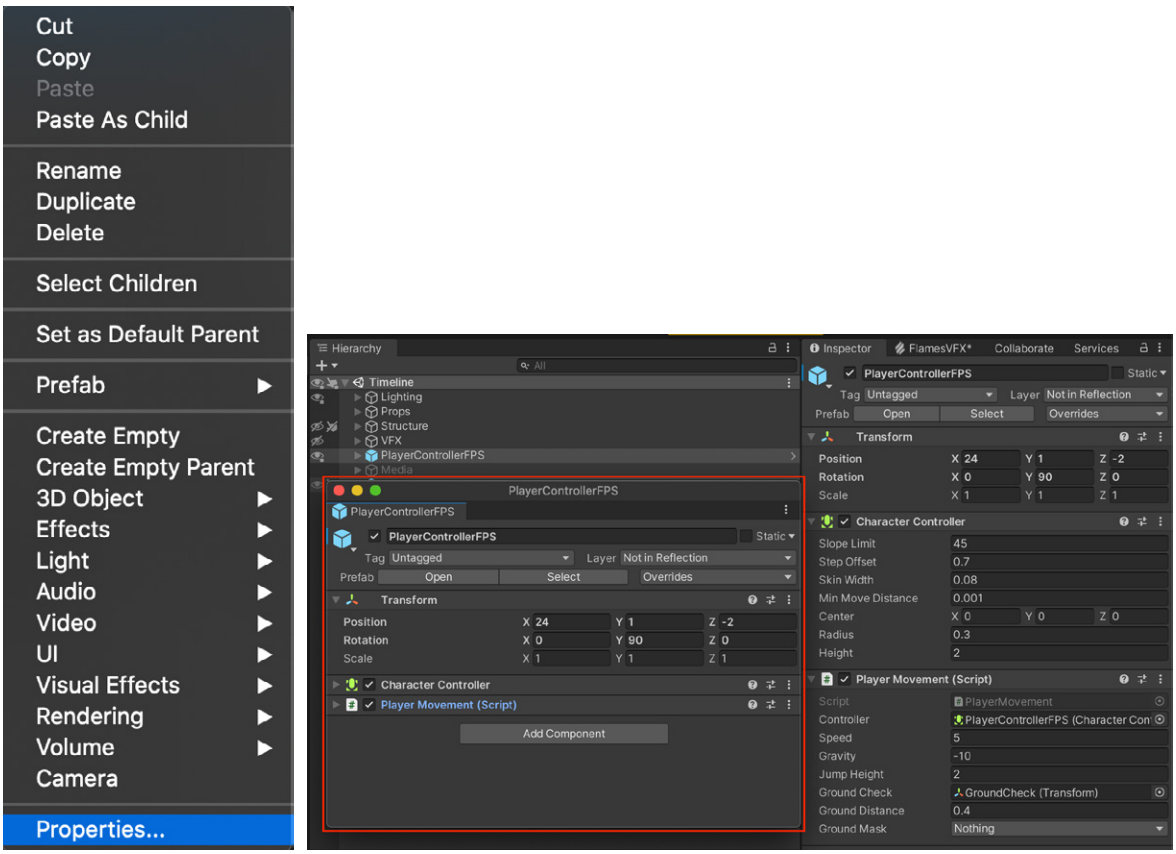
Command	Shortcut
View	Q
Move	W
Rotate	E
Scale	R
Rect	T
Transform	Y
Toggle Pivot Position	Z
Toggle Pivot Orientation	X

Основные сочетания клавиш редактора

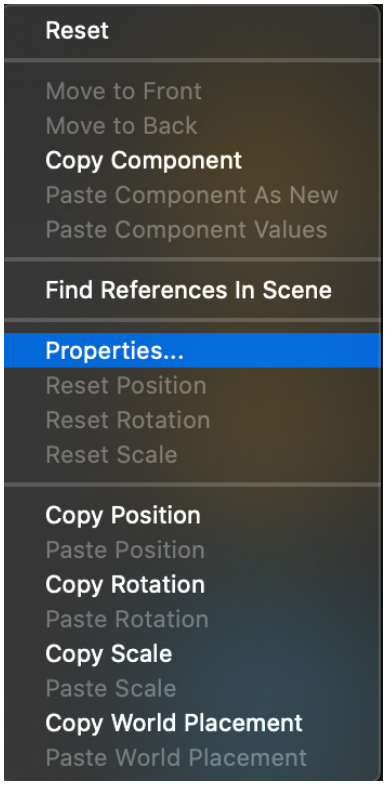
Работайте с несколькими окнами Inspector

Окно Focused Inspector позволяет одновременно открыть несколько окон Inspector и просматривать в каждом свойства определенного объекта GameObject, компонента или ассета. В таком окне всегда отображаются свойства того элемента, для которого оно было открыто, даже если в сцене выбран другой элемент.

Щелкните правой кнопкой мыши объект GameObject или компонент и выберите Properties. Откроется отдельное плавающее окно Inspector, которое, как и любое другое окно, можно перемещать, закреплять и изменять его размер.



Сравнение двух объектов GameObject в окне Focused Inspector



Если одновременно открыть несколько окон Focused Inspector, можно обращаться к нескольким объектам GameObject при редактировании сцены и удобно сравнивать их бок о бок.

Также можно открыть Focused Inspector только для определенного компонента объекта GameObject, чтобы сэкономить место на экране.

Настройте значения по умолчанию с помощью Presets

Пресеты позволяют настраивать и повторно использовать состояние компонентов или ассетов по умолчанию в окне Inspector. При создании пресета Unity сохраняет текущие настройки компонента или ассета в виде ассета с расширением .preset.

Затем эту конфигурацию можно применить к другим компонентам или ассетам того же типа.

Используйте пресеты для соблюдения стандартов или задания подходящих значений по умолчанию новым ассетам. Это помогает поддерживать единые стандарты в команде, чтобы часто упускаемые из виду настройки не снижали производительность проекта.

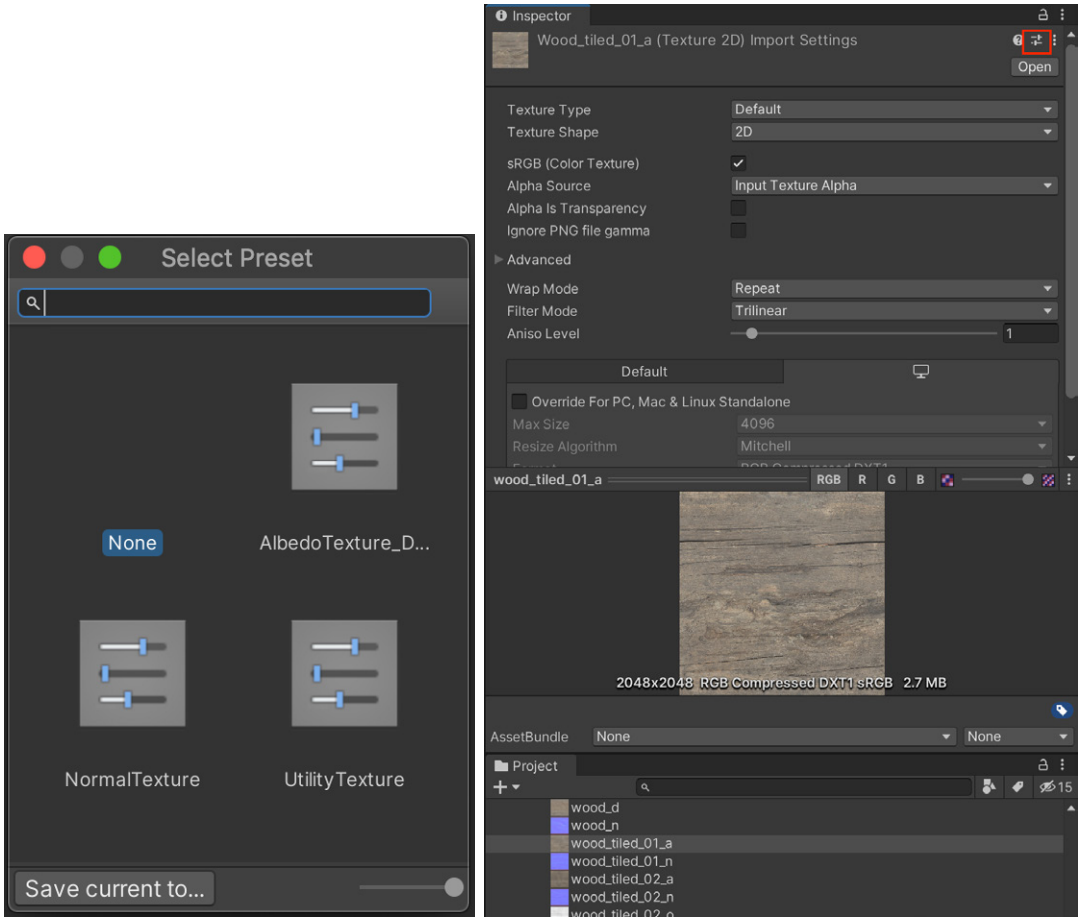
Значок Preset выделен красной рамкой.

Чтобы создать пресет:

1. Щелкните значок Preset в правом верхнем углу компонента.

2. Нажмите Save current to..., чтобы сохранить пресет как ассет.

3. Выберите один из доступных пресетов, чтобы загрузить соответствующий набор значений.



В этом примере пресеты содержат разные Import Settings для 2D-текстур в зависимости от назначения: альбедо, карта нормалей или служебная текстура.

Другие практические способы использования пресетов:

- Создание объекта GameObject со значениями по умолчанию: перетащите ассет пресета в окно Hierarchy. Будет создан новый объект GameObject с соответствующим компонентом, заполненным значениями пресета.

- Связь определенного типа с пресетом: в Preset Manager (Project Settings > Preset Manager) укажите для каждого типа один или несколько пресетов. После этого при создании нового компонента по умолчанию будут использованы значения указанного пресета.

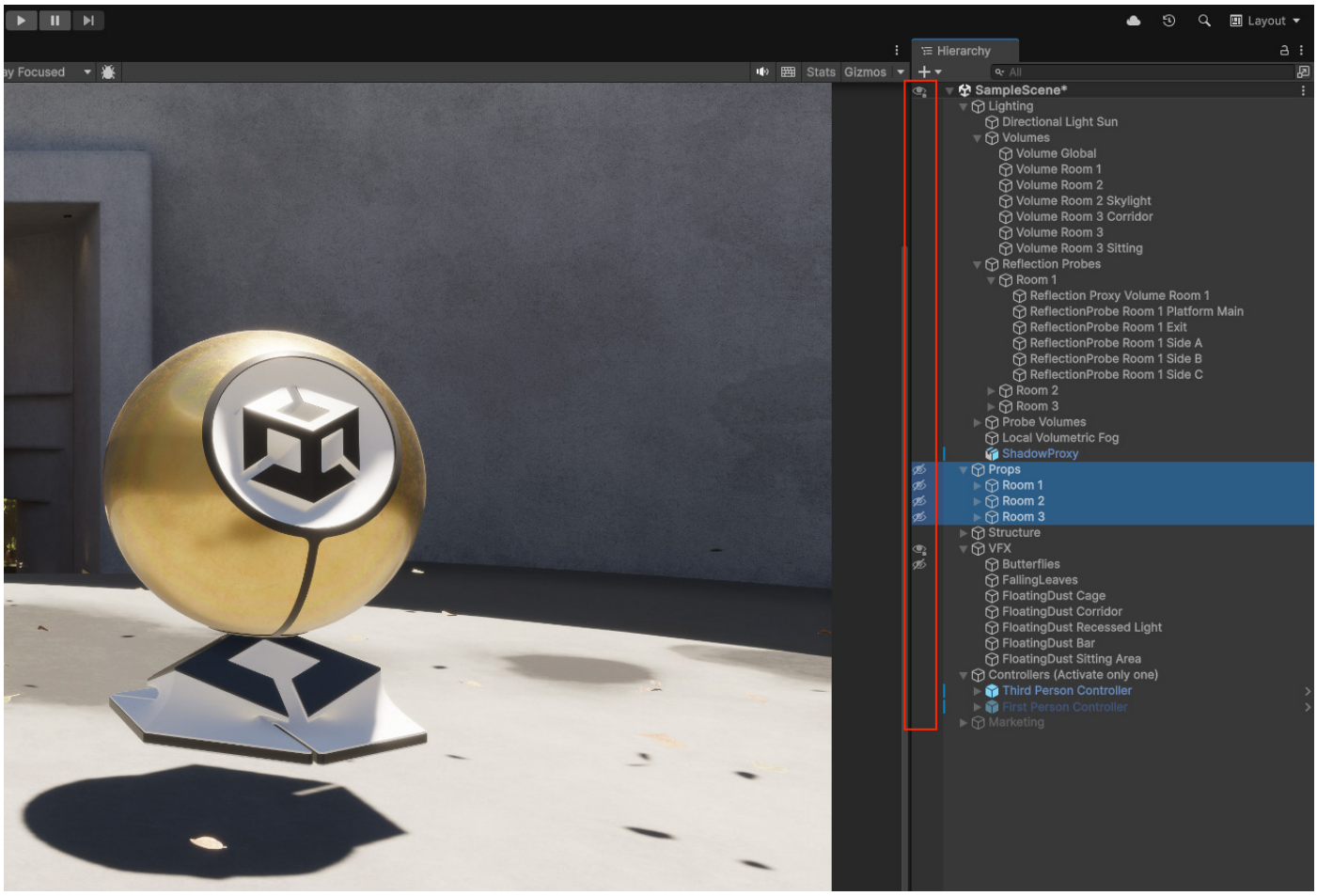
- Выбор из нескольких пресетов с помощью фильтров: можно создать несколько пресетов для одного типа и задать в Preset Manager фильтры, определяющие, какой пресет будет применяться.

- Сохранение и повторное использование настроек менеджеров: применяйте пресеты к окнам менеджеров, чтобы повторно использовать их настройки, например Tags and Layers или Physics. Пресеты сокращают время настройки следующего проекта.

Уберите лишнее из окна Scene с помощью Scene visibility

По мере усложнения сцены можно временно скрывать отдельные объекты, чтобы упростить выбор и редактирование объектов GameObject.

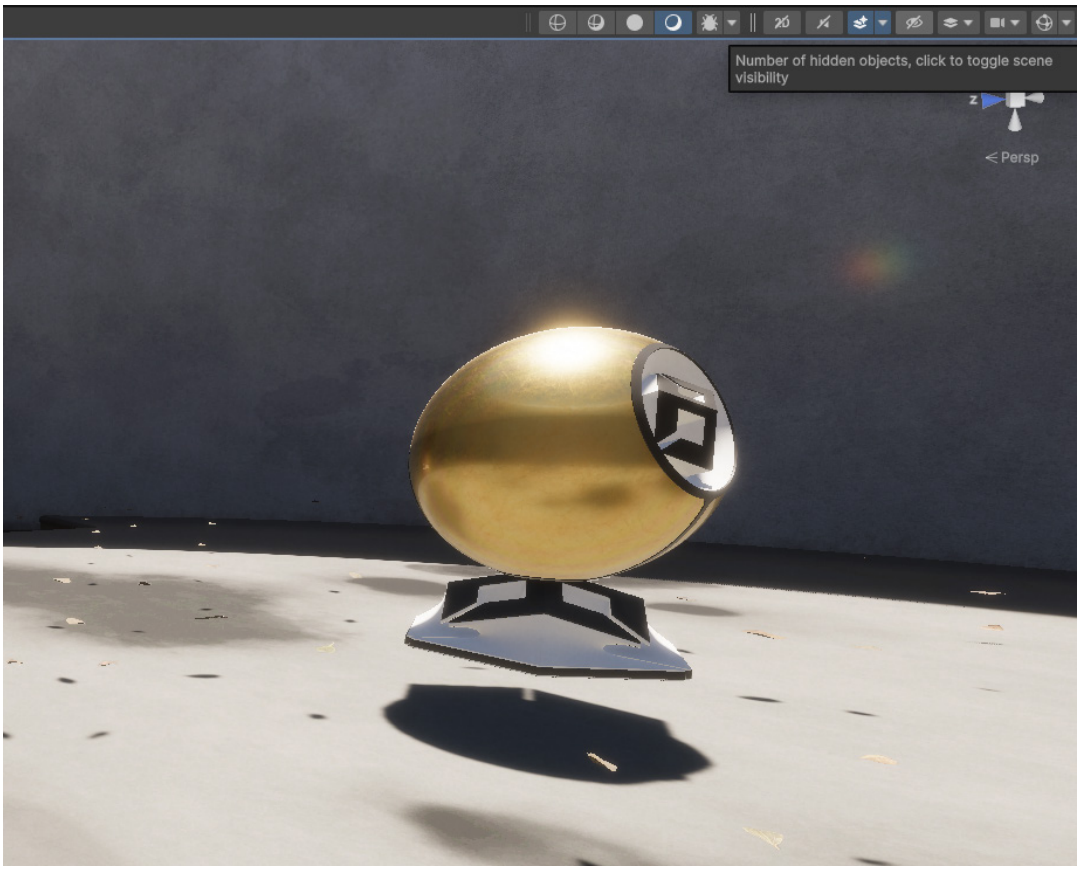
Вместо того чтобы деактивировать объекты GameObject, что может привести к непредвиденному поведению, используйте элементы управления Scene visibility для переключения их видимости в окне Scene. Щелкните значок глаза рядом с объектом GameObject в окне Hierarchy. Объект исчезнет из окна Scene, но его активное состояние и видимость в игре не изменятся.



Скрытие объектов в окне Scene с помощью элементов управления Scene visibility.

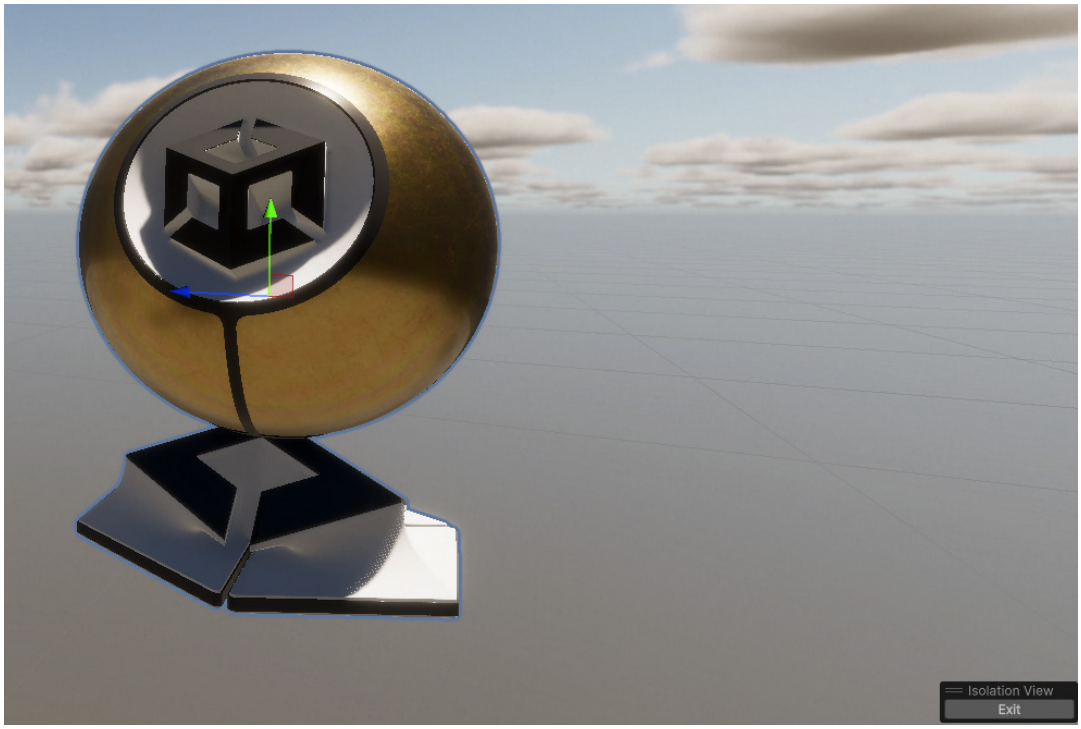
Обратите внимание: значки состояния в окне Hierarchy могут меняться в зависимости от того, скрыты родительские или дочерние объекты.

Значок	Статус
	Объект GameObject виден, но некоторые его дочерние объекты скрыты.
	Объект GameObject скрыт, но некоторые его дочерние объекты видны.
	Объект GameObject и его дочерние объекты видны; значок отображается только при наведении указателя на объект GameObject.
	Объект GameObject и все его дочерние объекты скрыты.



Включайте и отключайте панель управления окна Scene, чтобы переопределять глобальные настройки Scene visibility.

Для работы только с определенным объектом и его дочерними объектами можно также использовать Isolation View. Выберите объект GameObject в окне Hierarchy и нажмите Shift + H, чтобы включить или отключить этот режим. До выхода из Isolation View он переопределяет остальные настройки Scene visibility.



Isolation View позволяет редактировать объект GameObject, не отвлекаясь на остальные элементы.

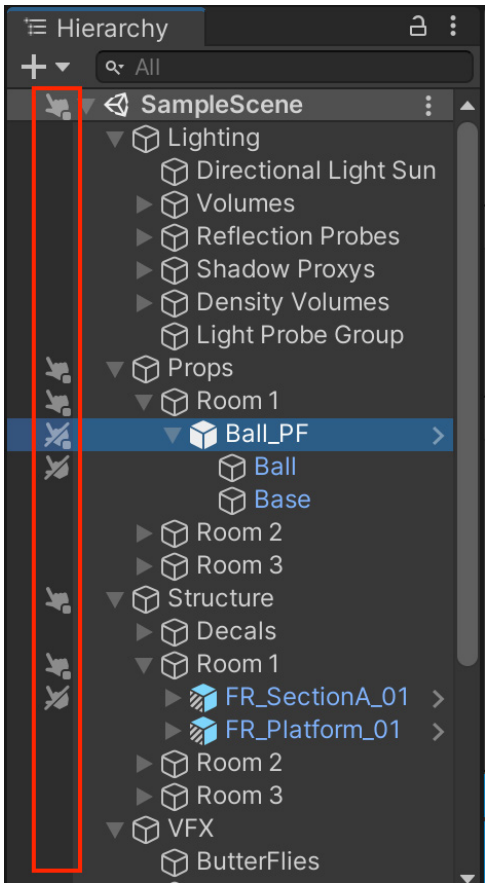
Помните: сочетание Shift + Пробел позволяет развернуть область просмотра на весь экран и скрыть остальные окна редактора.

В Unity можно работать с несколькими сценами. Это полезно при создании больших миров с потоковой загрузкой и для эффективного управления несколькими сценами во время выполнения. Данные можно запекать одновременно для нескольких сцен, включая карты освещения, данные NavMesh и данные отсечения по окклюзии. Кроме того, редактировать несколько сцен с помощью скриптов можно как в редакторе, так и во время выполнения.

Избегайте случайного выбора объектов в окне Scene

Unity позволяет запрещать выбор отдельных объектов GameObject в окне Scene, аналогично управлению Scene visibility.

На панели инструментов Scene visibility можно переключать состояние Pickability объекта GameObject и тем самым блокировать выбор определенных объектов GameObject в окне Scene. Это помогает избежать случайного выбора и редактирования не того объекта GameObject в больших и сложных сценах.



Pickability в окне Hierarchy

Поскольку Pickability можно переключать как для целой ветви, так и для отдельного объекта, некоторые объекты GameObject могут быть доступны для выбора, а их дочерние или родительские объекты могут быть недоступны. Различать эти состояния помогают следующие значки.

Значок	Статус
	Объект GameObject можно выбрать, но некоторые его дочерние объекты выбрать нельзя.
	Объект GameObject выбрать нельзя, но некоторые его дочерние объекты можно.
	Объект GameObject и его дочерние объекты можно выбрать; значок отображается только при наведении указателя на объект GameObject.
	Нельзя выбрать ни объект GameObject, ни его дочерние объекты.

Просматривайте ассеты прямо в окне Inspector

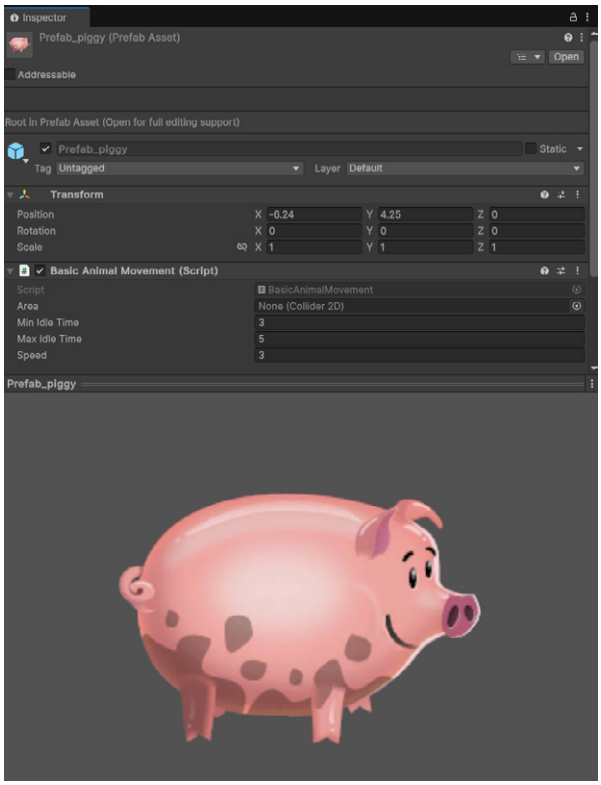
Asset Preview - панель в нижней части окна Inspector в Unity Editor, где можно просмотреть ассет или его миниатюру, например 3D-модель, материал или текстуру. Она позволяет быстро и интерактивно оценивать ассеты прямо во время работы в редакторе.

Для 3D-ассетов, таких как модели и префабы, можно щелкнуть область Asset Preview в нижней части окна Inspector и, не отпуская кнопку мыши, поворачивать объект, осматривая его с разных сторон. Так можно быстро проверить геометрию, материалы и ориентацию ассета, не добавляя его на сцену.

Если область Asset Preview слишком мала, измените ее размер, перетащив разделитель над ней в окне Inspector.

Для анимированных моделей Asset Preview нередко воспроизводит простую анимацию по умолчанию, если она есть в ассете. Это позволяет быстро оценить движение и риггинг.

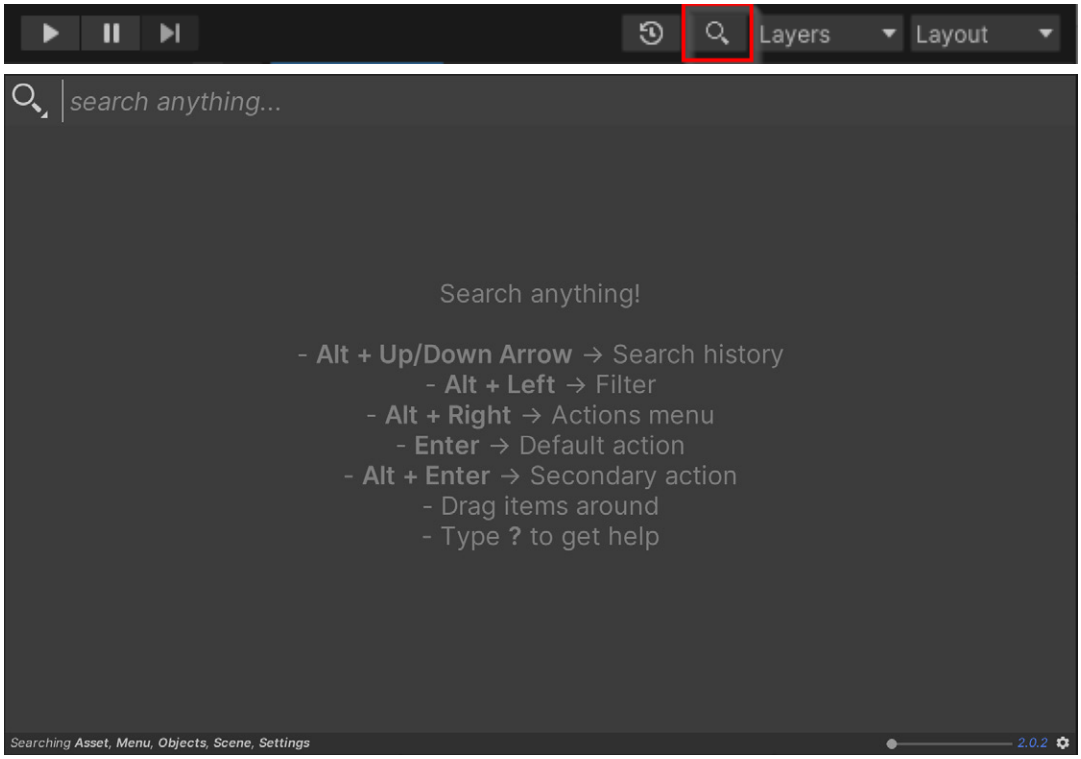
Unity также может показывать миниатюры, то есть уменьшенные версии Asset Preview, прямо в окне Project. Для этого переключите окно Project в режим Two Column Layout и настройте ползунок масштаба миниатюр внизу. Так ассеты проще просматривать и упорядочивать визуально.



Панель Asset Preview в демопроекте Happy Harvest

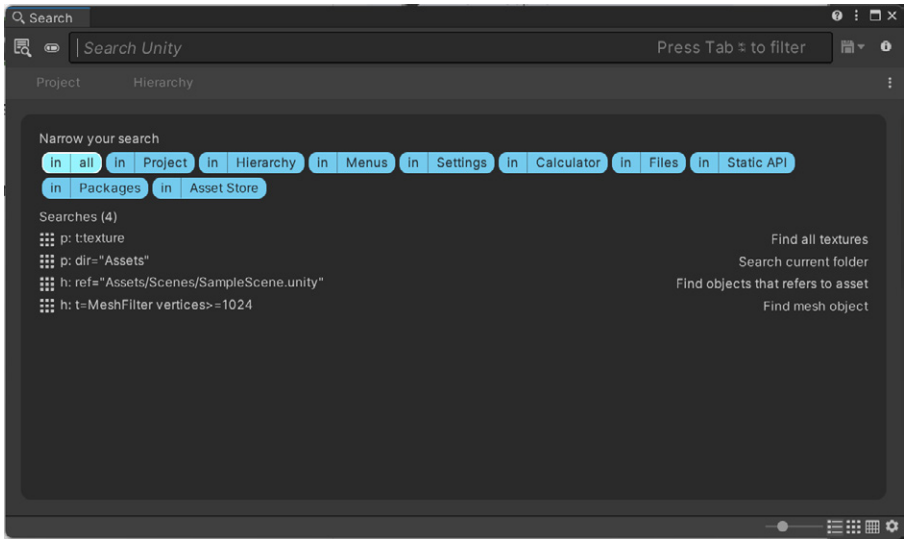
Ускорьте поиск в проектах

В Unity можно эффективно искать ассеты, объекты сцены, пункты меню, пакеты, API, настройки и многое другое. Помимо поиска в окнах Hierarchy и Project, можно нажать значок поиска в главной строке меню или использовать Ctrl + K в Windows либо Cmd + K в macOS.

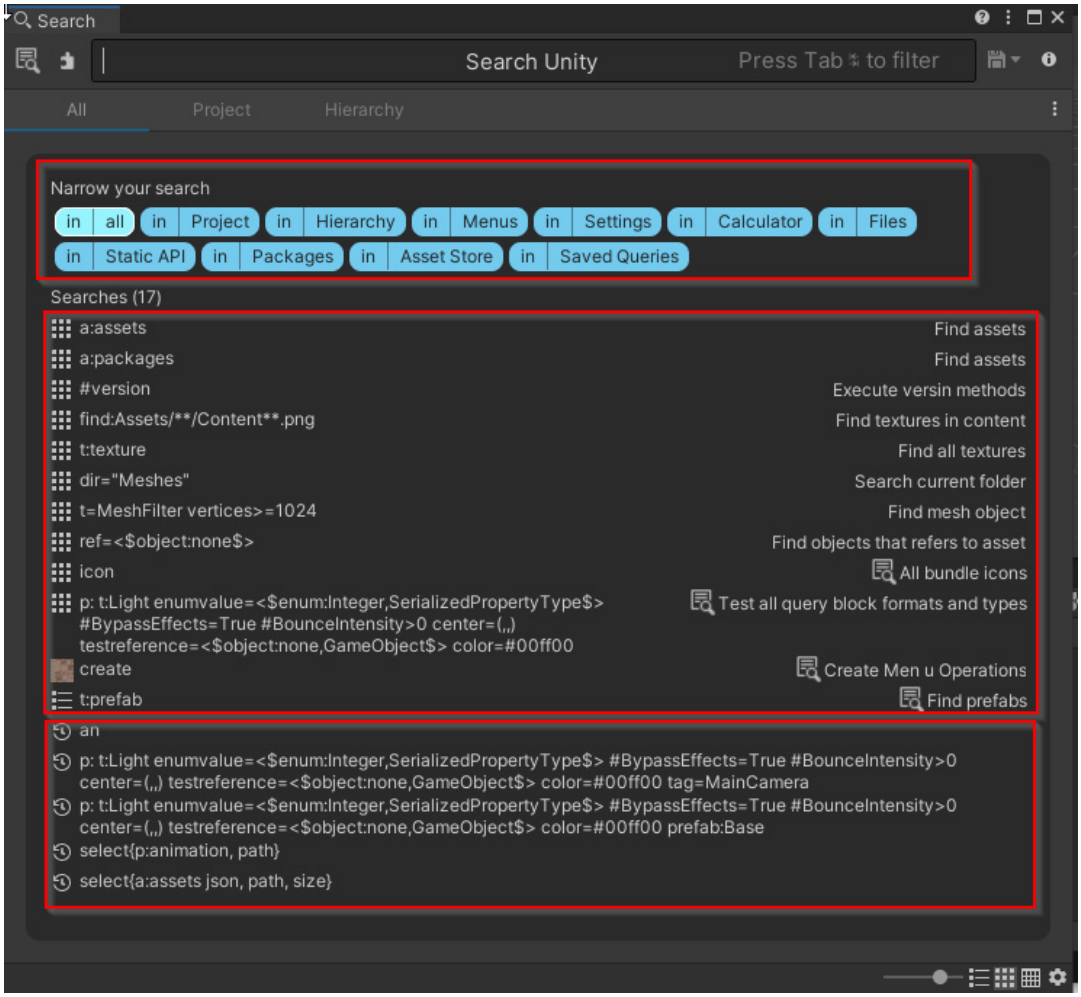


Используйте сочетание клавиш или меню Help, чтобы открыть QuickSearch.

Откроется окно Search, в котором удобно фильтровать результаты.



Окно Search



В средней части окна отображаются запросы, доступные для выбранной области поиска. Щелкните запрос, чтобы выполнить его. По именам можно искать с учетом типа: выберите Type в раскрывающемся списке или используйте сокращенный синтаксис t:, например t:scene для поиска по всем сценам или t:texture для поиска по всем текстурам.

Кроме того, окно Search поддерживает несколько способов отображения объектов: компактный или крупный список, а также сетку со значками разных размеров. Объекты можно вывести и в виде таблицы.

Подробнее о поиске внутри Unity и за ее пределами рассказывается в руководстве QuickSearch.

Создавайте собственные фильтры поиска с помощью Query Builder

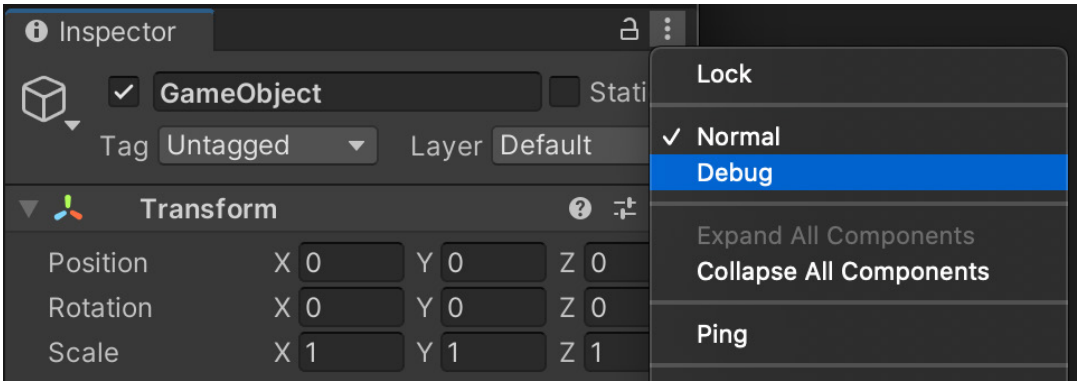
При работе со сложными поисковыми запросами Query Builder упрощает изучение проекта. Его можно включить переключателем конструктора, то есть кнопкой с пазлом рядом с полем Search Field.

Query Builder

Экономьте время, создавая и сохраняя собственные запросы для типовых задач. Например, если вам часто приходится искать префабы по тегам или неиспользуемые ассеты, создайте повторно используемый запрос и автоматизируйте эту работу. Query Builder теперь встроен непосредственно в редактор и больше не поставляется как отдельный пакет, что упрощает рабочие процессы команды.

Отображайте отладочные данные в окне Inspector

Для Inspector каждого объекта GameObject можно переключаться между режимами Normal и Debug. Нажмите кнопку More Items (⋮), чтобы открыть контекстное меню, и выберите нужный режим. В режиме Debug в Inspector отображаются все сериализованные поля, включая закрытые, скрытые ссылки и внутренние данные Unity. Это помогает при поиске неисправностей и позволяет изучать компоненты за пределами их обычного открытого интерфейса.

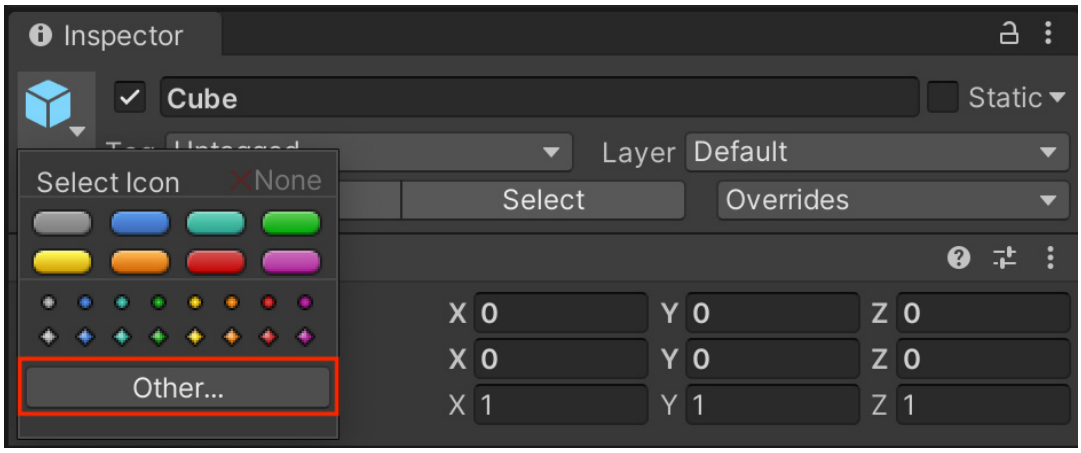


Режим Debug в Inspector

Пользовательские гизмо и значки для визуальной отладки

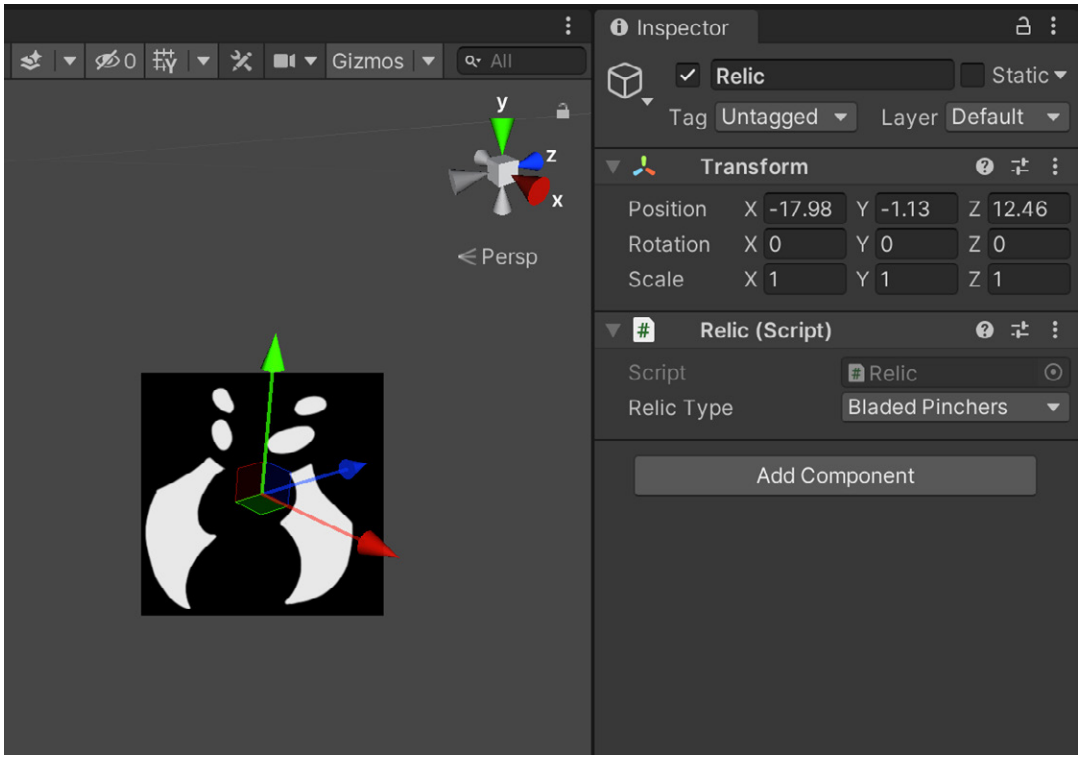
Гизмо - визуальные элементы, которые появляются в окне Scene и помогают находить и различать объекты GameObject во время разработки. Они особенно полезны для отображения элементов, которые не отрисовываются в игре, например триггеров, точек появления и скриптов.

Значок объекта GameObject можно изменить в меню Select Icon. Выберите Other, чтобы задать собственный значок.



Для переключения гизмо используйте раскрывающийся список в Inspector.

Гизмо можно создавать программно и делать интерактивными. Например, с помощью гизмо можно обозначить объем или область действия пользовательского компонента.



В этом примере скрипт изменяет гизмо в зависимости от выбранного объекта.

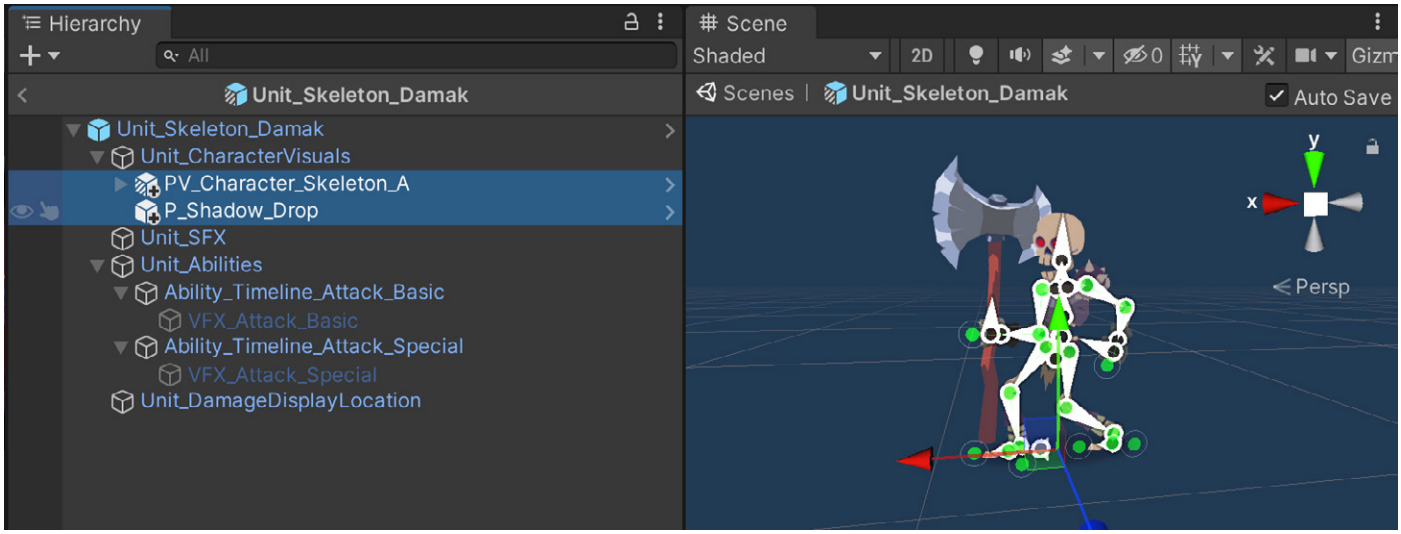
Используйте диалоговое окно Gizmos на панели управления окна Scene, чтобы включать и отключать отдельные гизмо или сразу все.

Примеры использования см. в руководстве «Создание пользовательских гизмо для разработки». Также ознакомьтесь с API Gizmos и Handles.

Используйте вложенные префабы и варианты префабов

Префабы позволяют сохранять и повторно использовать полностью настроенные объекты GameObject, а также гибко и эффективно создавать сцены. Возможности префабов этим не ограничиваются: их можно вкладывать друг в друга, создавать варианты и использовать для создания экземпляров объектов GameObject во время выполнения.

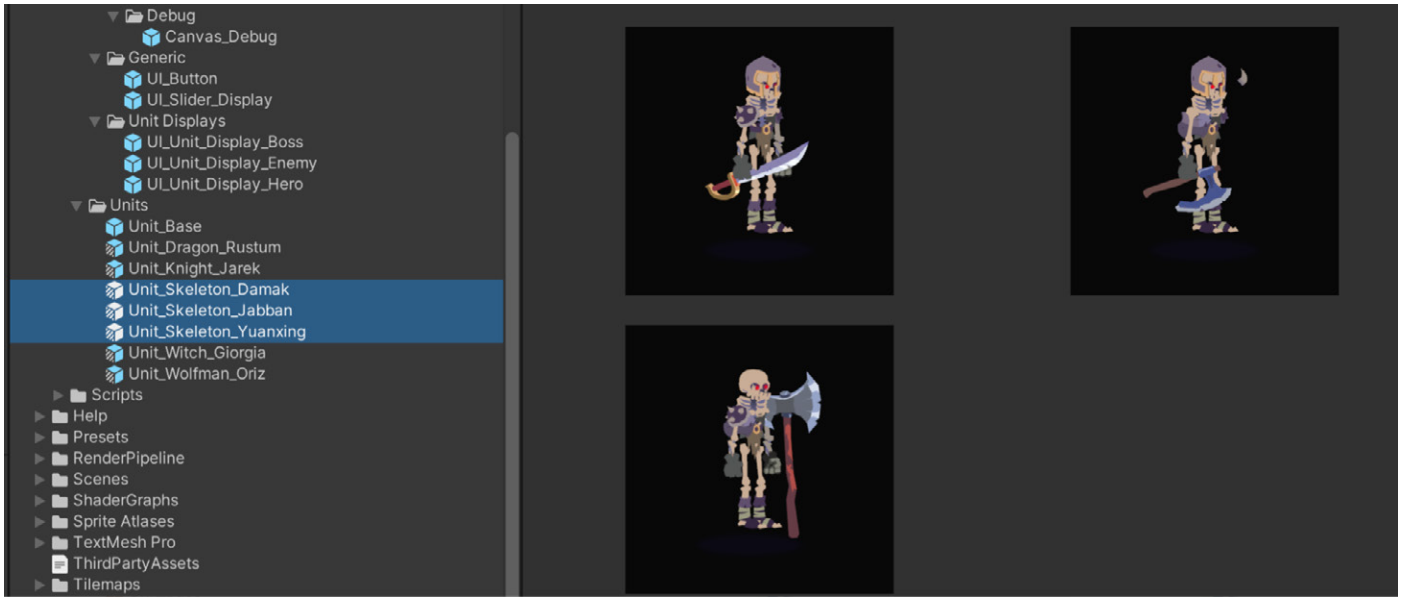
Вложенные префабы позволяют делать одни префабы дочерними по отношению к другим. Например, крупный префаб здания можно собрать из небольших префабов комнат и мебели. Такой подход помогает распределить работу над ассетами между художниками и разработчиками, чтобы они одновременно работали над разными частями контента.



Пример вложенных префабов в демопроекте Dragon Crashers

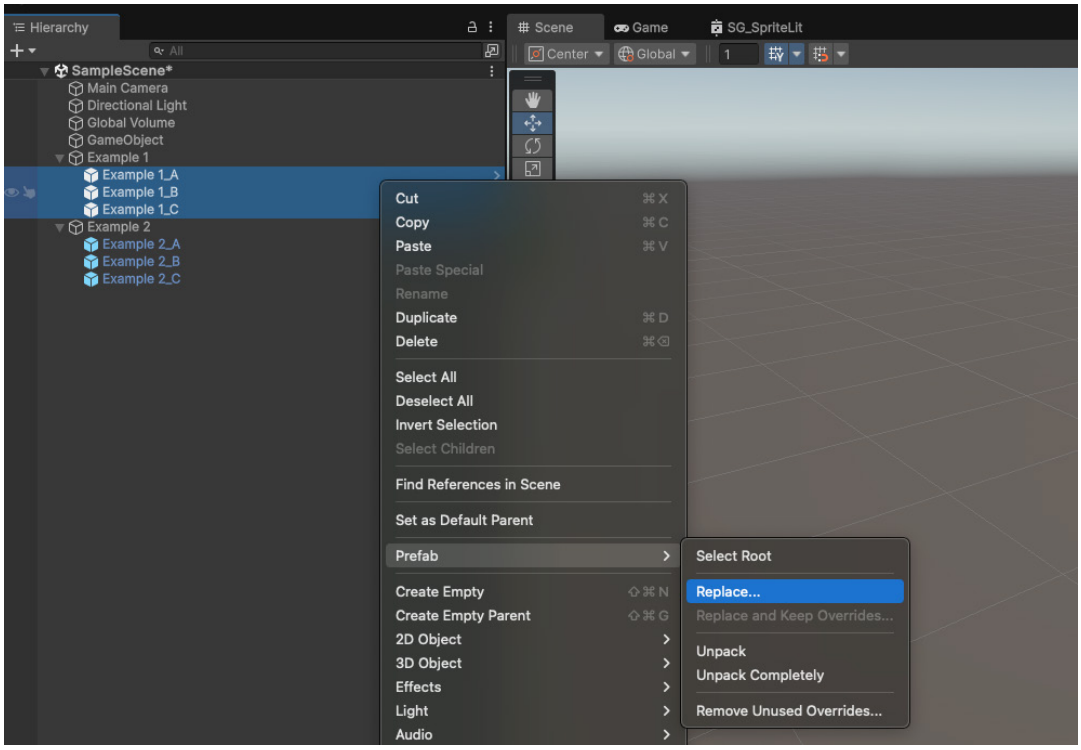
Вариант префаба создается на основе другого префаба по принципу, похожему на наследование в объектно-ориентированном программировании. В варианте можно переопределить отдельные части, не затрагивая оригинал. Все переопределения можно в любой момент удалить, вернувшись к базовому префабу.

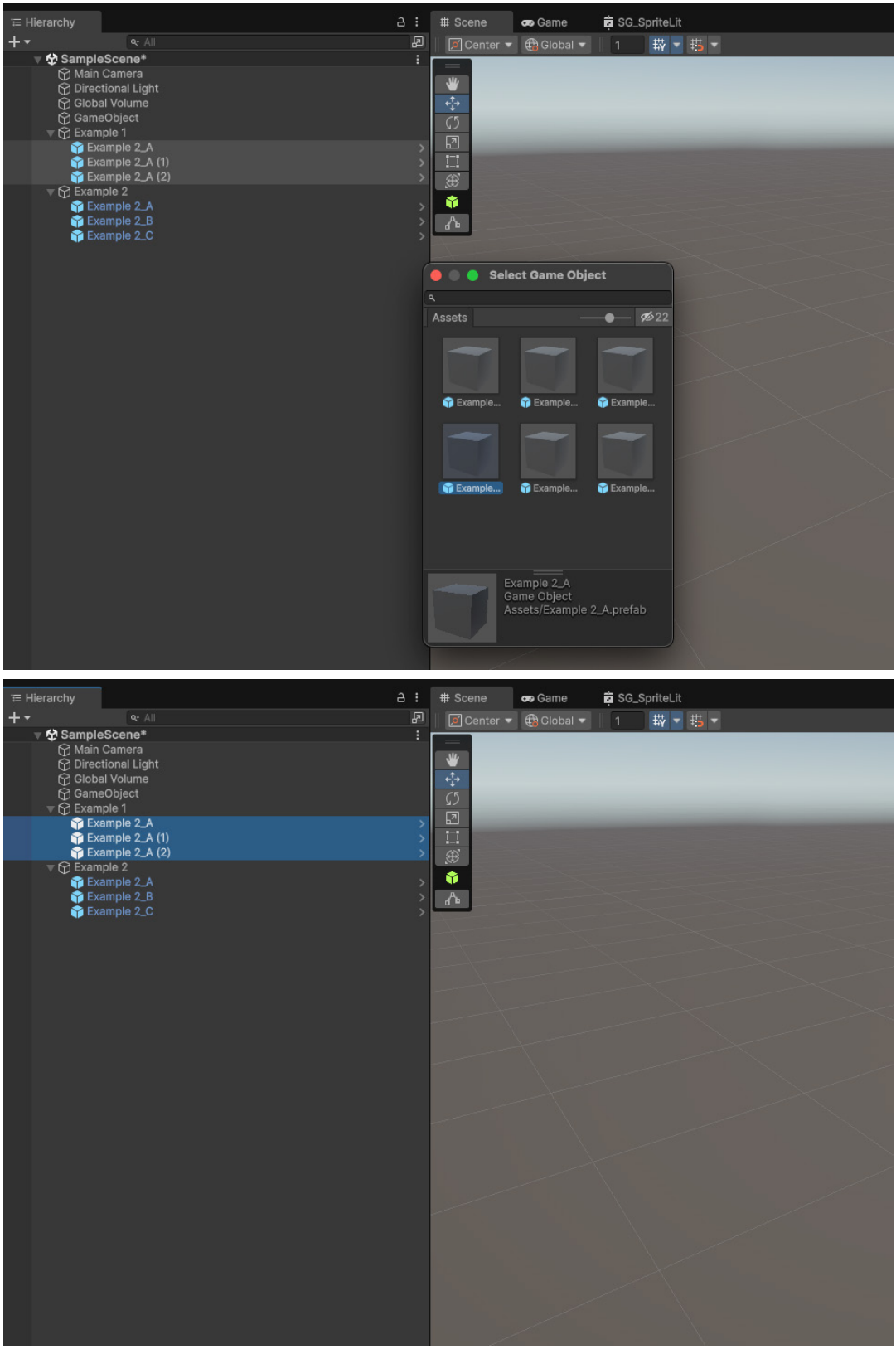
Чтобы изменить сразу все варианты, внесите изменения непосредственно в базовый префаб.



Варианты префабов в демопроекте Dragon Crashers

Чтобы заменить один или несколько префабов на сцене другим, удерживайте Ctrl в Windows или Cmd в macOS и перетащите новый префаб на заменяемые префабы.

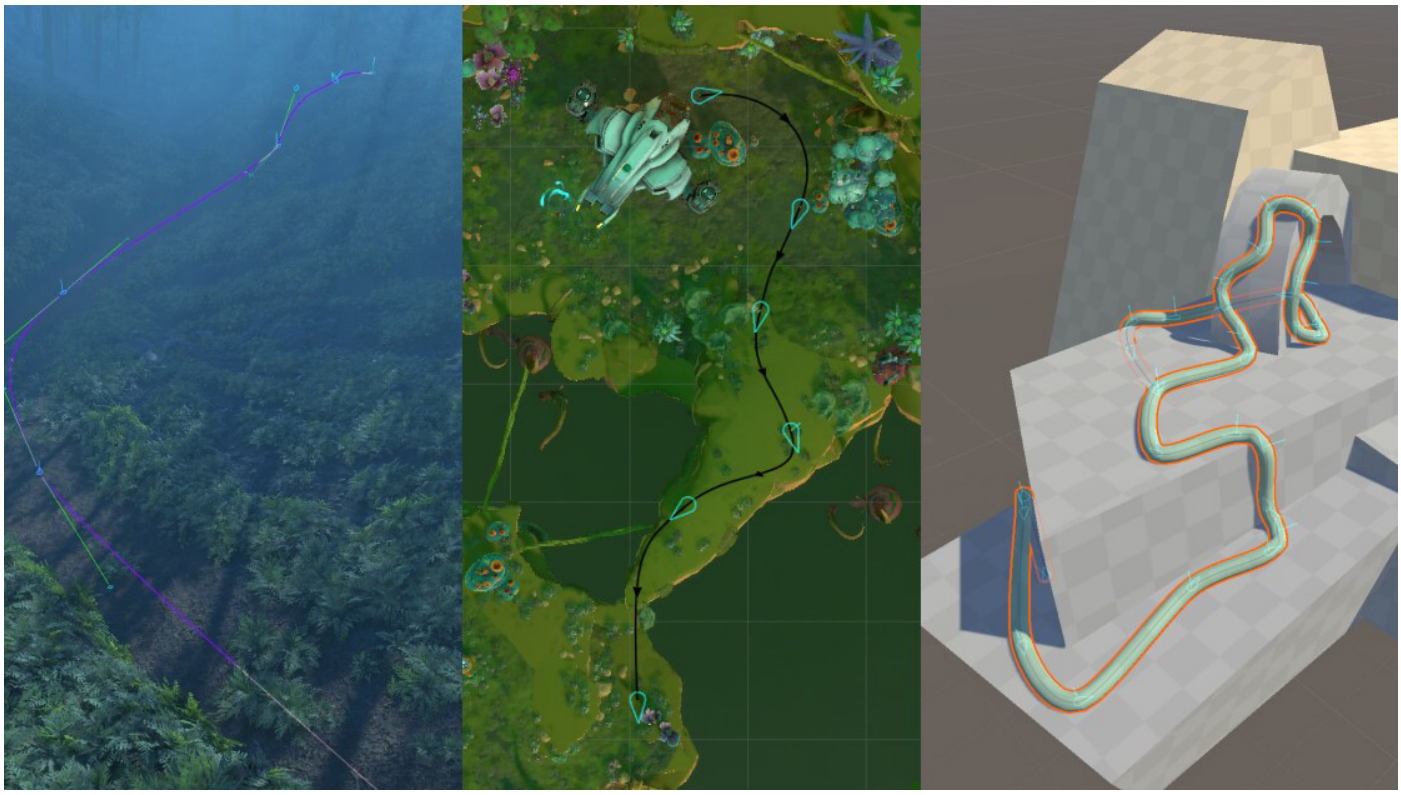




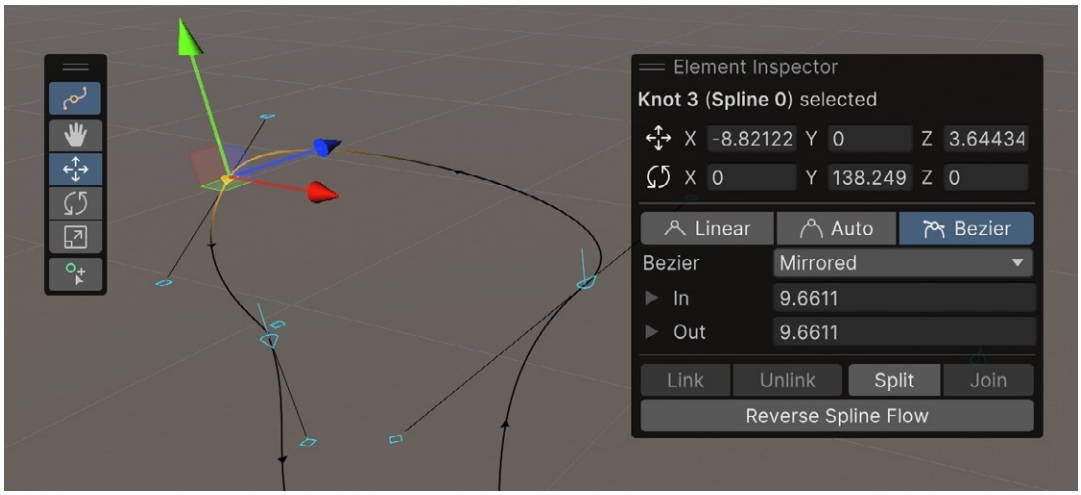
Замена префаба в окне Hierarchy

С лёгкостью создавайте криволинейные пути с помощью пакета Splines

Пакет Splines позволяет создавать и редактировать сплайновые траектории прямо в редакторе. Сплайны подходят для построения плавных криволинейных путей, которыми можно задавать дороги, реки, рельсы, траектории камеры и любые другие визуальные или игровые элементы, основанные на движении по заданному пути.



Примеры применения сплайнов слева направо: дорога через лес, траектория анимации и трубчатые меши. После установки пакета другие примеры его возможностей доступны на странице Splines в Package Manager.



Манипуляторы и элементы управления сплайнами в Unity напоминают инструменты векторного и 3D-рисования в известных DCC-приложениях.

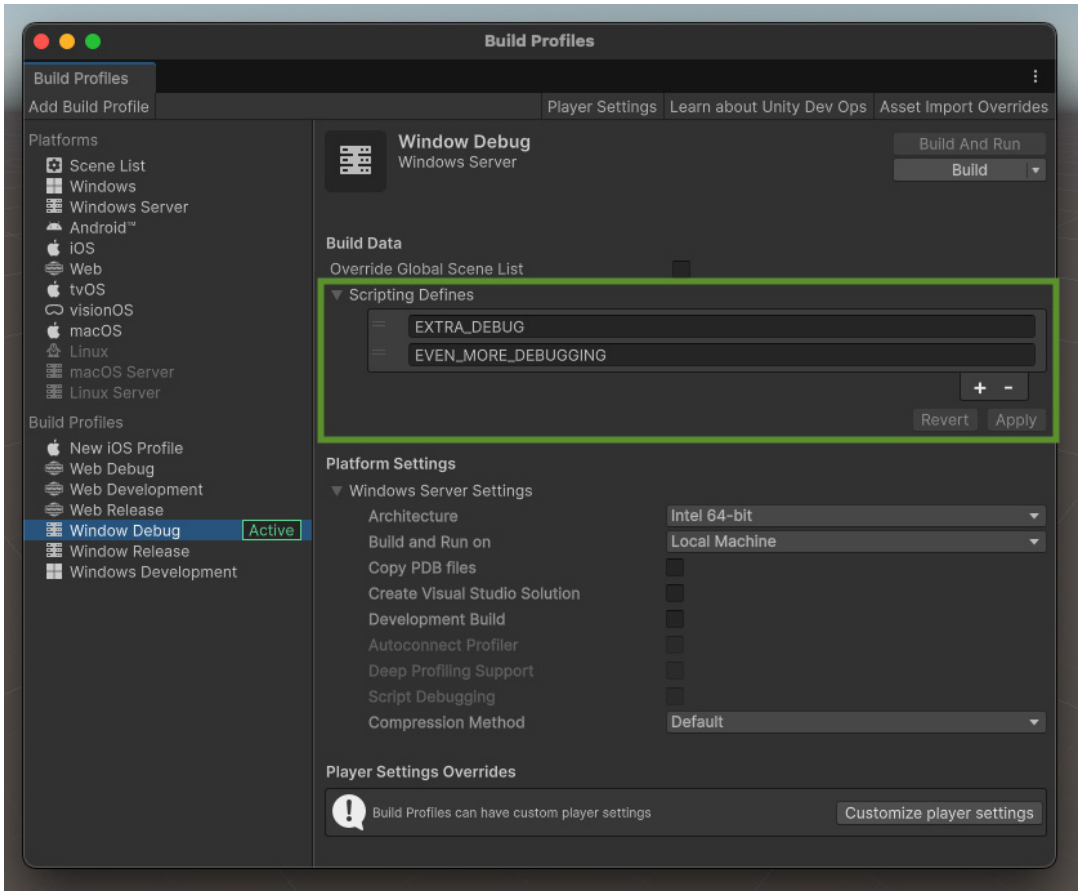
После создания сплайна программисты и художники могут применять его самыми разными способами. Например, программист может считывать точки сплайна через API и использовать их в игровой логике.

Посмотрите видеоруководство «Начало работы с пакетом Splines»: в нём показано, как решать типовые задачи со сплайнами, анимировать положение и вращение объекта GameObject вдоль сплайна и размещать вдоль него экземпляры префабов для создания окружения.

Настраивайте сборки с помощью Build Profiles

Build Profiles - новый рабочий процесс настройки сборок в Unity 6. Он позволяет создавать для каждой платформы несколько конфигураций с разными параметрами.

Создавайте в Unity 6 собственные профили в Build Profiles, чтобы удобно задавать параметры сборки для разных платформ и сценариев, например для отладки, тестирования или выпуска. Между такими конфигурациями можно быстро переключаться, оптимизируя рабочий процесс.



Используйте параметры Scripting Defines и Player Settings Overrides в Build Profiles, чтобы задавать настройки конкретной сборки, например включать средства отладки или выбирать заставку, и легко делиться ими с командой через систему контроля версий.

Дополнительные ресурсы:

- Что нужно знать о Build Profiles в Unity 6

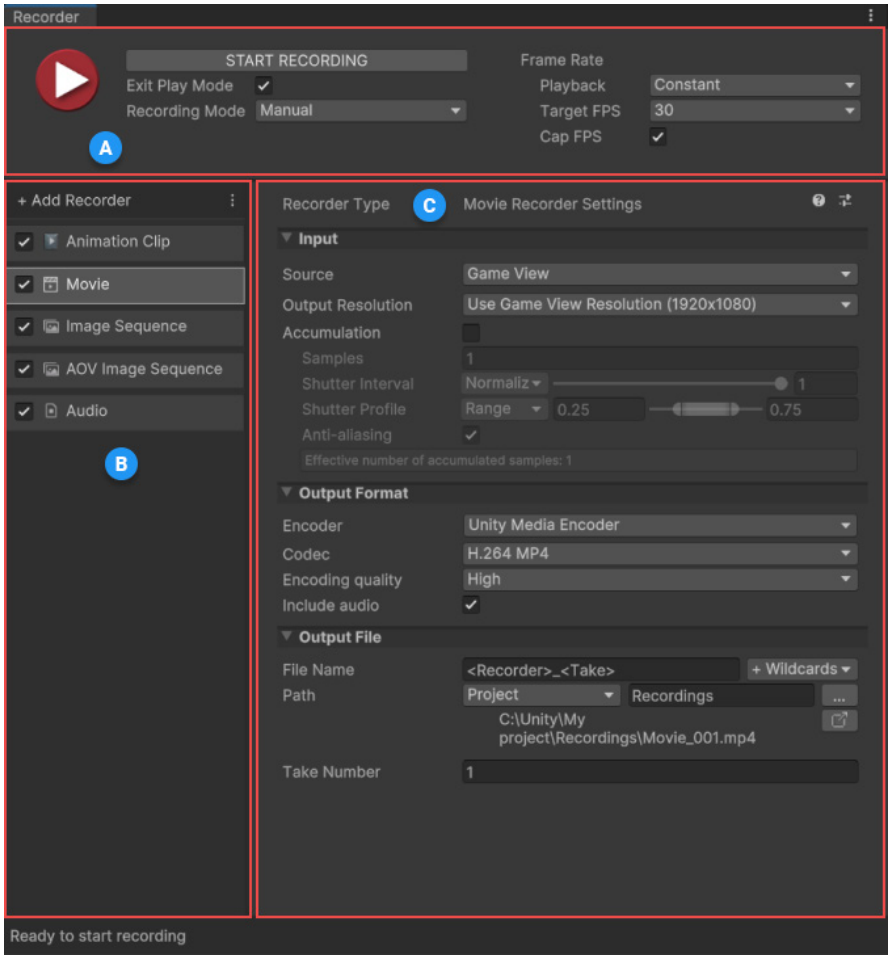
- Все, что нужно знать о Build Profiles в Unity 6 | Unite 2024

Записывайте плавное видео игрового процесса с помощью пакета Recorder

Пакет Unity Recorder позволяет записывать и экспортировать данные проекта в режиме Play Mode. Игровой процесс, кинематографические сцены и другие выполняемые в игре последовательности можно сохранять как видеофайлы, последовательности изображений или анимационные данные.

Окно Recorder находится в главном меню Unity Editor в разделе Window. При открытии Recorder Unity восстанавливает параметры последнего сеанса записи, поэтому запись можно быстро повторить или скорректировать настройки.

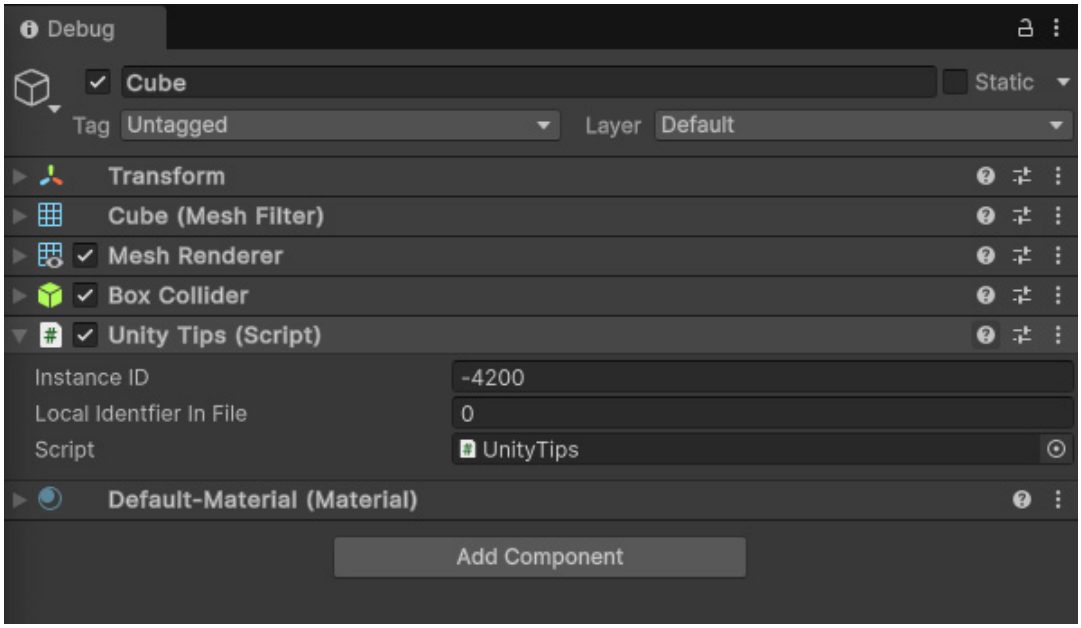
Можно записывать изображения и видео с прозрачным фоном. При определенных настройках, зависящих от используемого конвейера рендеринга и параметров Recorder, Unity Recorder сохраняет альфа-канал и прозрачность.



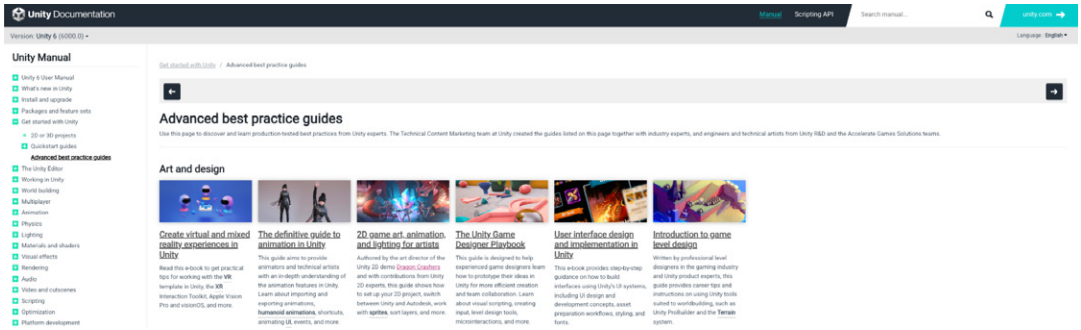
Окно Unity Recorder

Советы по работе в редакторе

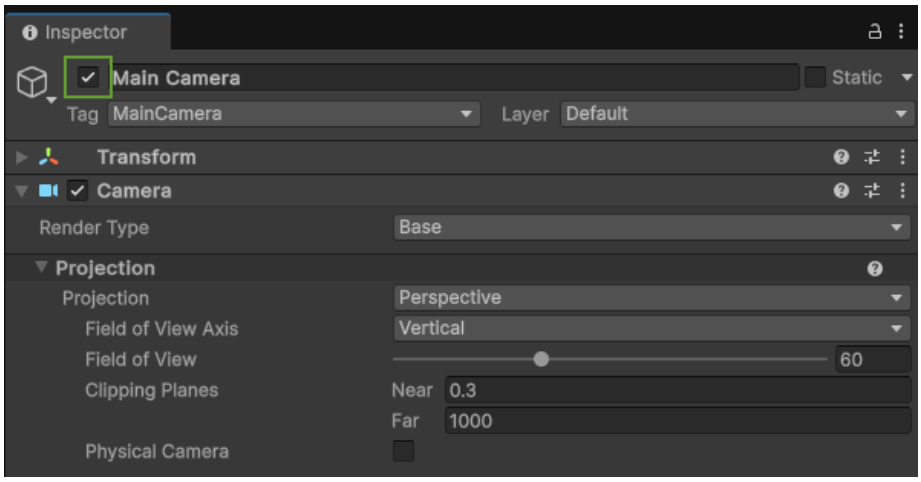
1. Используйте атрибут [HelpURL], чтобы связать значок вопросительного знака в Inspector с документацией или другими ресурсами. Если щелкнуть этот значок в Inspector, указанный URL откроется в браузере пользователя по умолчанию.




```
using UnityEngine;

[HelpURL ("https://docs.unity3d.com/Manual/best-practice-guides.html")]
public class UnityTips : MonoBehaviour
{    // Start is called once before the first execution of Update after the MonoBehaviour is created
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```



2. Ежедневно экономьте несколько щелчков мыши. Удерживайте Shift + Alt + A (Windows) или Shift + Option + A (macOS), чтобы активировать или деактивировать выбранный в данный момент объект GameObject.



Активация и деактивация выбранного объекта GameObject

3. Unity позволяет настроить схему нумерации объектов GameObject. Она находится в Project Settings на вкладке Editor. Здесь можно задать правила именования, число разрядов и отступ перед номером экземпляра.

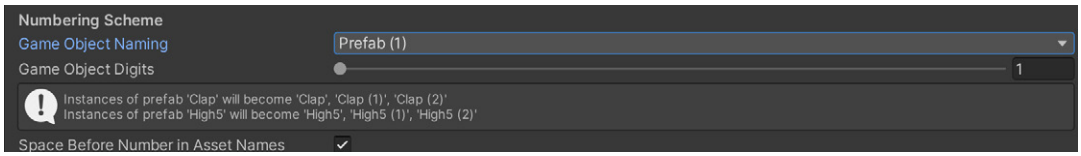


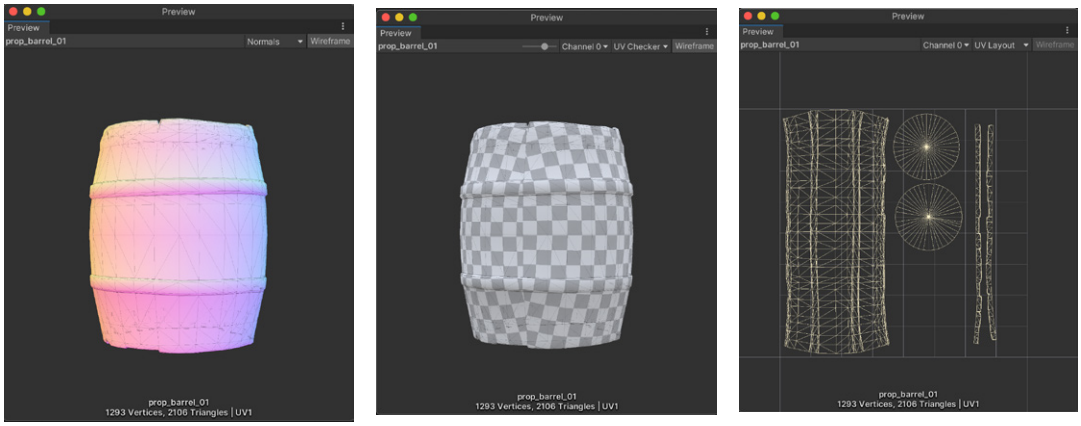
Схема нумерации

4. Вырезайте и вставляйте объекты GameObject в окне Hierarchy. В контекстном меню также доступна команда Paste As Child.

Команда Paste As Child

5. Нажмите F, чтобы сфокусировать окно Scene на выбранном объекте. В режиме Play Mode нажмите Shift + F, чтобы закрепить вид за выбранным движущимся объектом GameObject.

6. Отображайте UV-развёртку, нормали, касательные и другие сведения о Mesh в области предварительного просмотра окна Inspector.

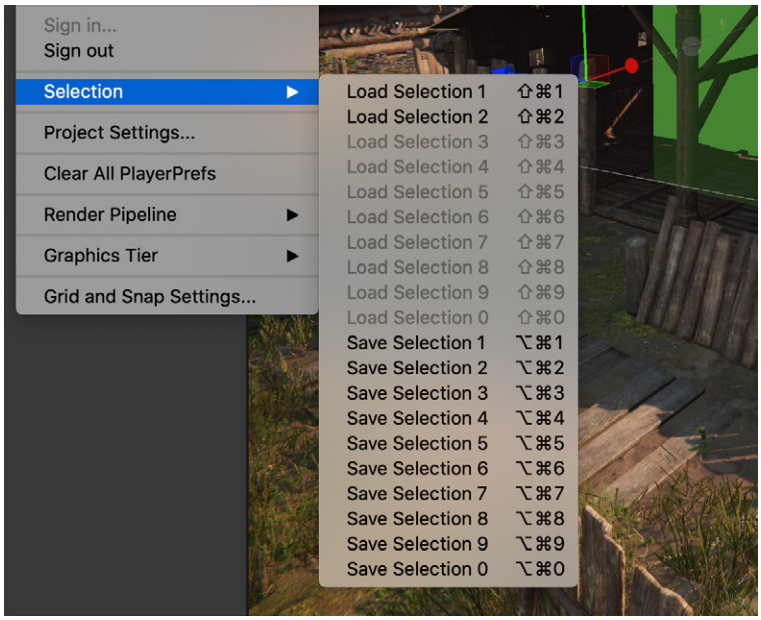


Предварительный просмотр в окне Inspector

7. В меню Layers отключайте видимость слоёв, например UI, которые могут заслонять содержимое окна Scene. Заблокируйте слой, чтобы случайно не изменить его состояние.

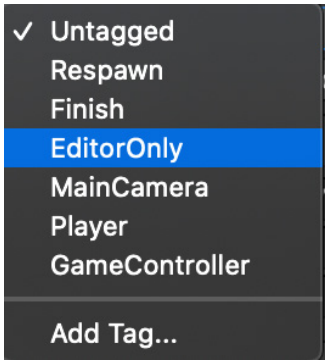
Переключение видимости и редактирование слоёв

8. Если вам часто приходится выбирать в сцене одни и те же объекты, сочетания клавиш в меню Edit > Selection позволяют быстро сохранять и загружать набор выделенных объектов.



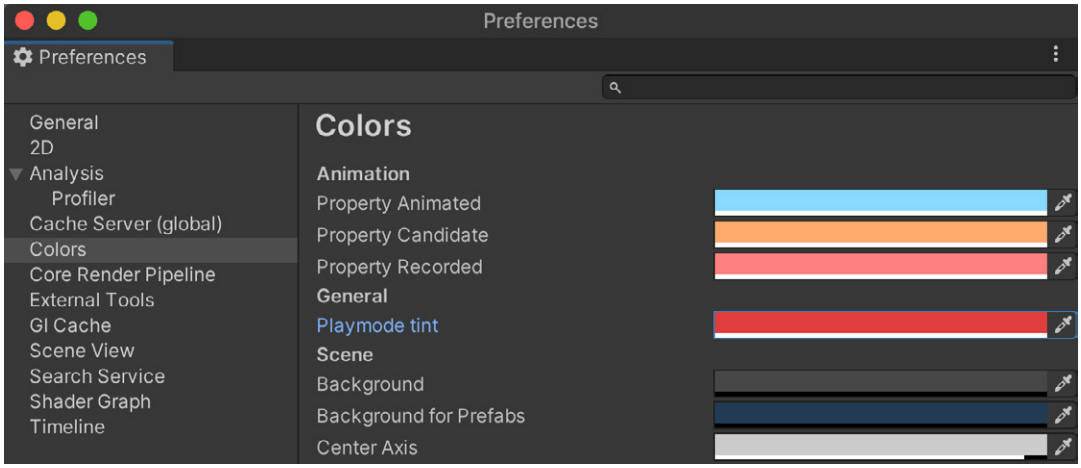
Загрузка и сохранение наборов выделенных объектов

9. Пометьте тегом EditorOnly объекты GameObject, которые не должны присутствовать в сборке приложения.



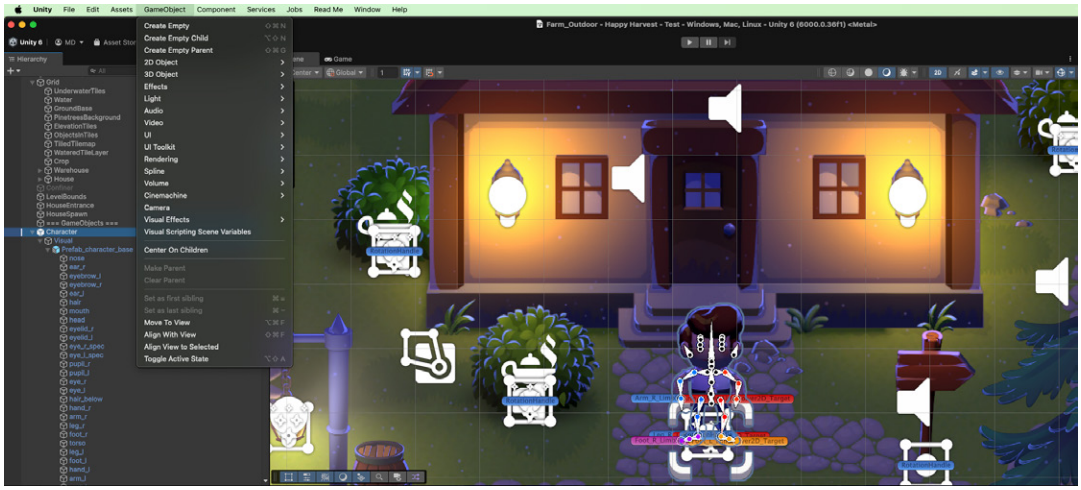
Тег EditorOnly

10. Измените цвета интерфейса редактора через Unity > Preferences > Colors, чтобы быстрее находить нужные элементы UI или объекты. Настройте Play mode tint: этот оттенок будет наглядно напоминать, что активен режим Play Mode, и поможет не потерять изменения, которые вы хотели сохранить перед выходом из него.



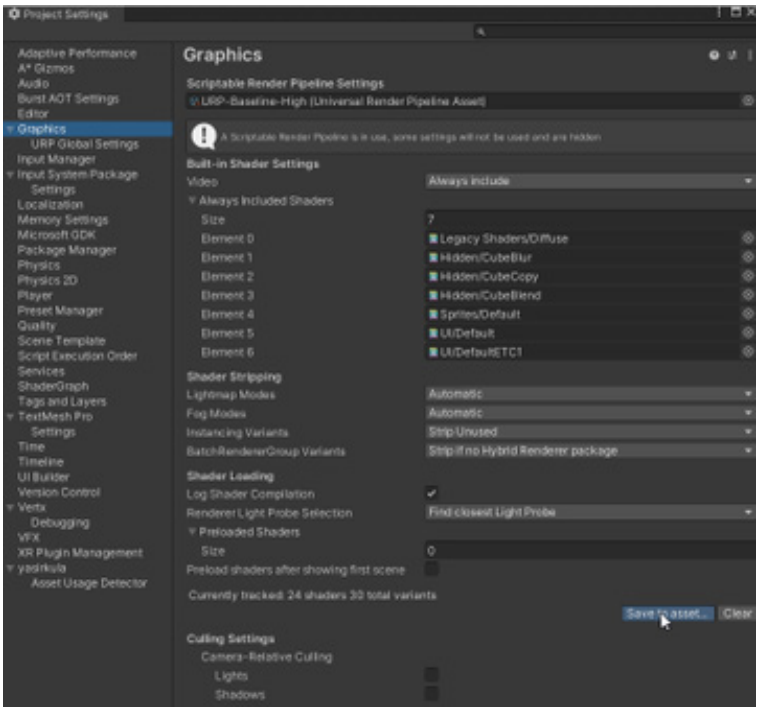
Параметр Play mode tint

11. При настройке камер выберите GameObject > Align With View, чтобы совместить игровую камеру с камерой окна Scene. Для обратного совмещения выберите Align View to Selected: камера окна Scene будет выровнена по другой камере, выбранной в окне Hierarchy.



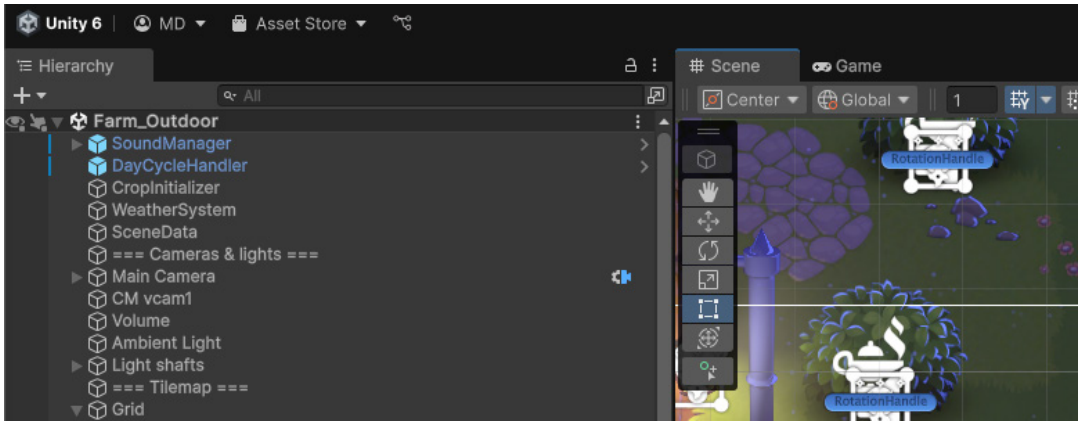
Команда Align With View

12. Можно создать список вариантов шейдеров, которые редактор использует в сцене. Откройте Edit > Project Settings > Graphics. В разделе Shader Loading проверьте количество шейдеров и их вариантов. Нажмите Save to asset..., чтобы создать ассет коллекции вариантов шейдеров.



Окно Graphics

13. Дважды щёлкните любую вкладку, например Project, Scene или Game, чтобы развернуть её на весь экран редактора.



Двойной щелчок по вкладке Scene

Вкладка Scene после двойного щелчка

Дополнительные ресурсы

Ещё больше коротких советов по повышению продуктивности вы найдёте в следующих материалах:

- Ускорьте рабочие процессы в Unity с помощью этих сочетаний клавиш
- Масштабируемые художественные ассеты: 6 советов по улучшению рабочих процессов
- Главные советы по написанию скриптов в Unity 2022 LTS

2D

Спрайты

В 2D-проекте визуальное представление создаётся с помощью спрайтов. Они могут использовать разные ассеты текстур, поэтому при неправильно настроенном батчинге может потребоваться много вызовов отрисовки.

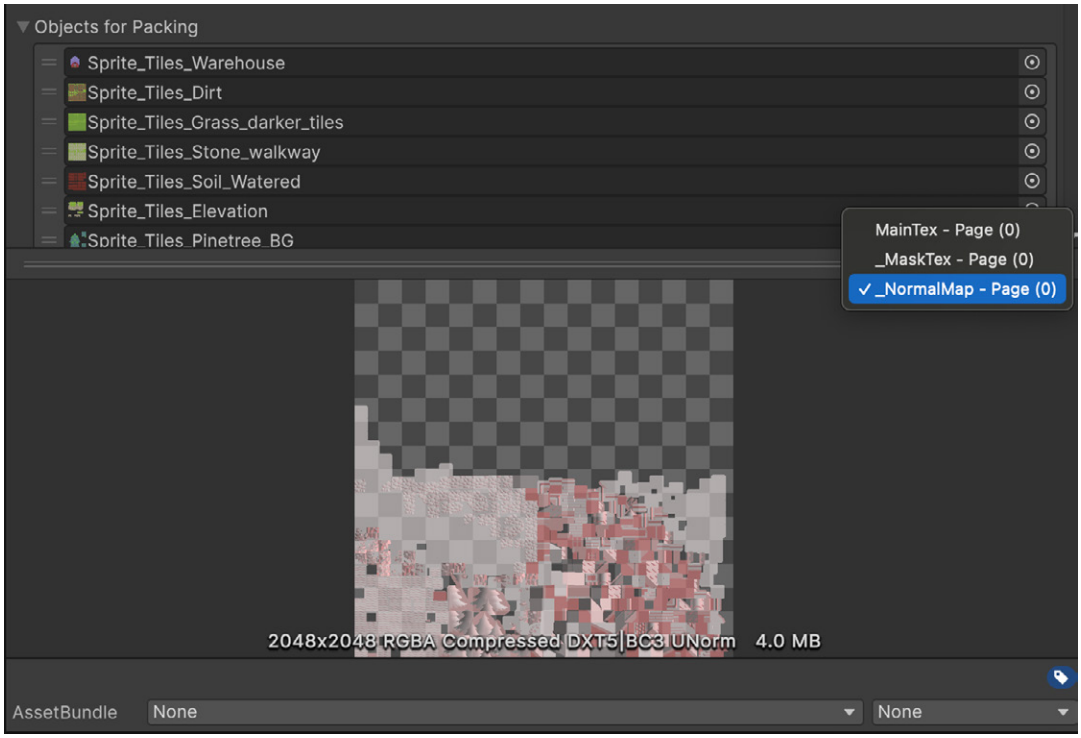
Для оптимизации ресурсов используйте Sprite Atlas (Asset > Create > Sprite Atlas): он позволяет объединять в один батч спрайты, использующие один материал и упакованную текстуру.

Избегайте дублирования ассетов при упаковке спрайтов

В 2D URP спрайты могут содержать карты нормалей и карты-маски, которые также стоит упаковать в Sprite Atlas. Это важно для производительности и качества: такая упаковка позволяет рациональнее использовать память, сократить число вызовов отрисовки и ускорить загрузку.

Если вы добавляете в Sprite Atlas целые папки, убедитесь, что в них находятся только обычные спрайты или текстуры альбедо. Карты нормалей и карты-маски автоматически попадут в соответствующие атласы, которые можно выбрать в раскрывающемся списке. Если же поместить такие карты в добавляемые папки, они без необходимости окажутся и в основном атласе, заняв в нём место.

Чёткое разделение атласов карт нормалей и карт-масок позволяет независимо настраивать для них параметры сжатия. Например, разрешение карт нормалей можно уменьшить вдвое, а обычные спрайты оставить в полном разрешении либо подобрать другие параметры с учётом целевой платформы.



Три типа атласов при упаковке 2D-спрайтов

Структурируйте папки, чтобы поддерживать единообразие

Продуманная структура папок для 2D-спрайтов упрощает поиск ассетов и управление ими, помогает контролировать использование памяти благодаря группировке ассетов по моменту загрузки, чётко разделяет разные элементы игры и даёт другие преимущества.

Ниже приведён типичный вариант структуры папок для проекта Unity:

- Sprites/Characters/[CharacterName]/Animations
- Sprites/UI/[MenuType]/Elements
- Sprites/Environment/[LevelName]/Background
- Sprites/Effects/[EffectType]
- Sprites/Common/SharedElements

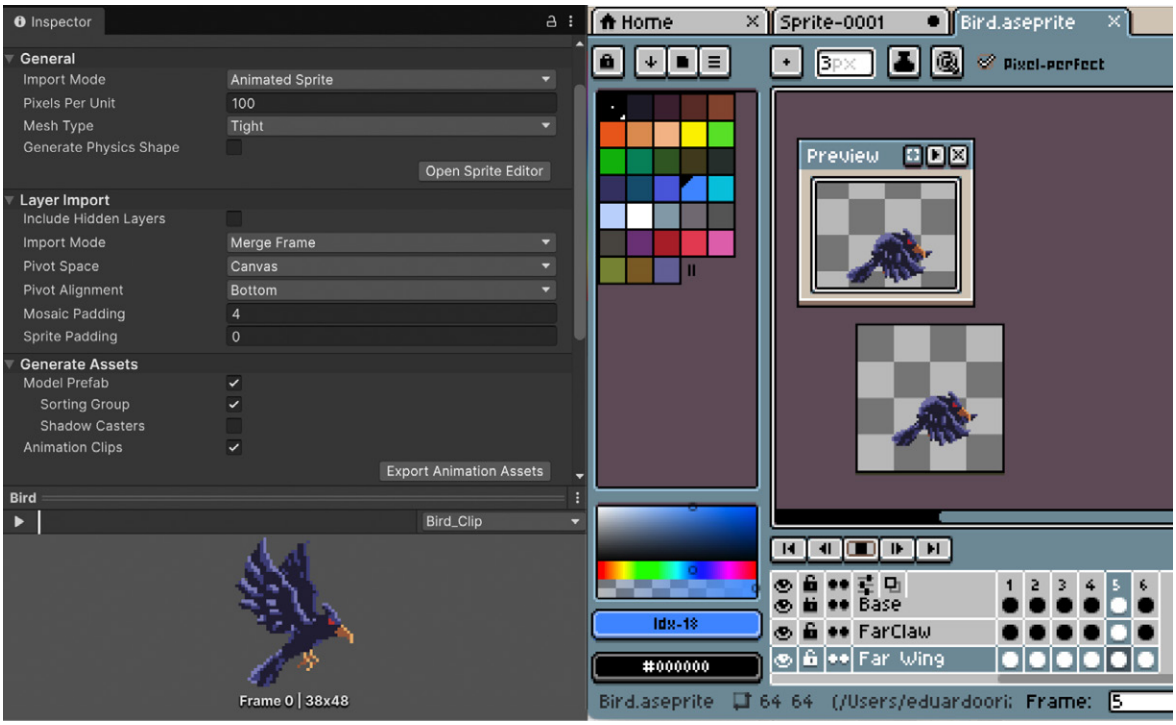
Импортируйте пиксельную графику прямо в Unity с помощью Aseprite Importer

Aseprite - редактор спрайтов и инструмент для создания пиксельной графики, ставший отраслевым стандартом среди DCC-приложений для пиксельных художников.

С Unity Aseprite Importer не нужно по отдельности экспортировать сотни спрайтов в формате .PNG: файл .aseprite можно добавить непосредственно в проект.

Работайте прямо в Aseprite, нажимайте Save и сразу просматривайте изменения в Unity. Aseprite Importer поддерживает перетаскивание файлов .aseprite, автоматически создаёт и настраивает готовые к использованию объекты GameObject и анимационные клипы, а также полностью интегрируется с набором 2D-функций Unity.

В Unity 6 появилась автоматическая генерация ассетов Tilemap из файлов .aseprite с тайлмапами. Изменения исходного файла автоматически переносятся во все производные ассеты.

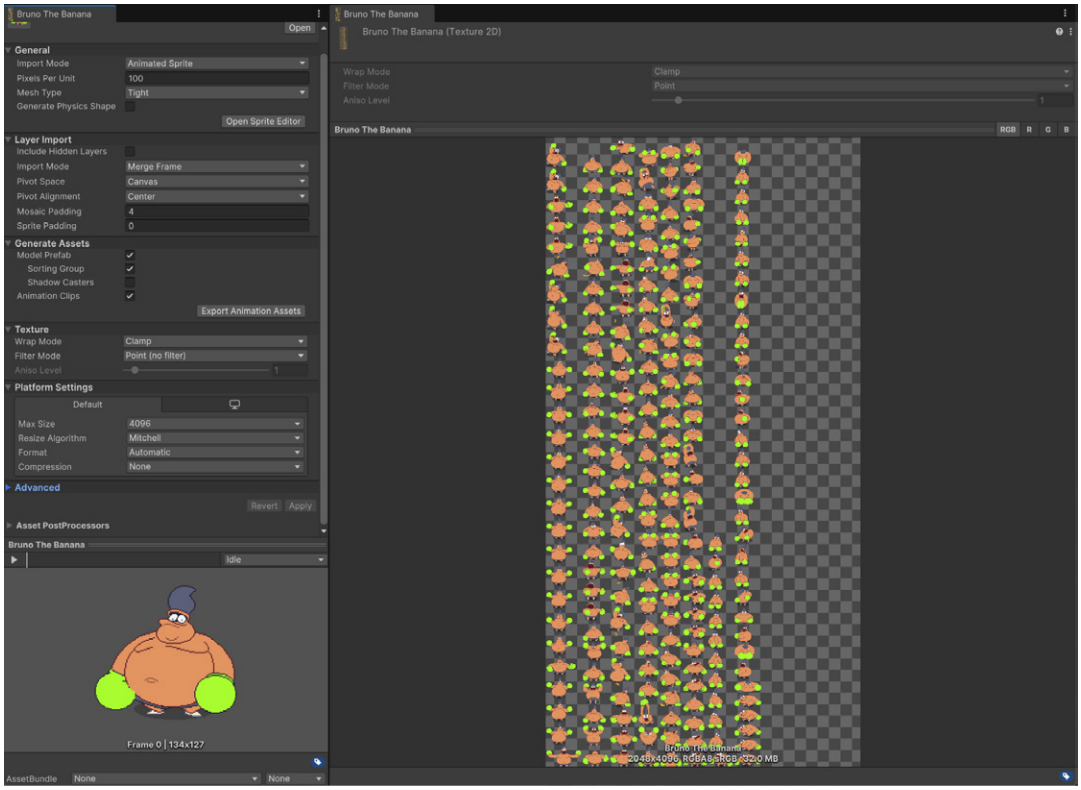


Aseprite Importer в окне Project в Unity: в этом фрагменте доклада Unite 2024 «Знакомьтесь: новые рабочие процессы 2D и улучшения на основе ИИ» показана работа Aseprite Importer

Импортируйте покадровую анимацию из Aseprite в Unity

Один из самых полезных советов по работе с Aseprite Importer в Unity: перед импортом тщательно организуйте файлы Aseprite с помощью Tags и Slices. Эти средства позволяют корректно и автоматически обрабатывать анимации и нарезать изображения на спрайты прямо в редакторе.

Например, определяйте и организуйте анимации с помощью тегов вроде Idle и Walk, а области спрайта или отдельные части, такие как голова и оружие, отмечайте с помощью Slices. Это упрощает импорт и экономит время.



Анимация, импортированная из Aseprite в Unity

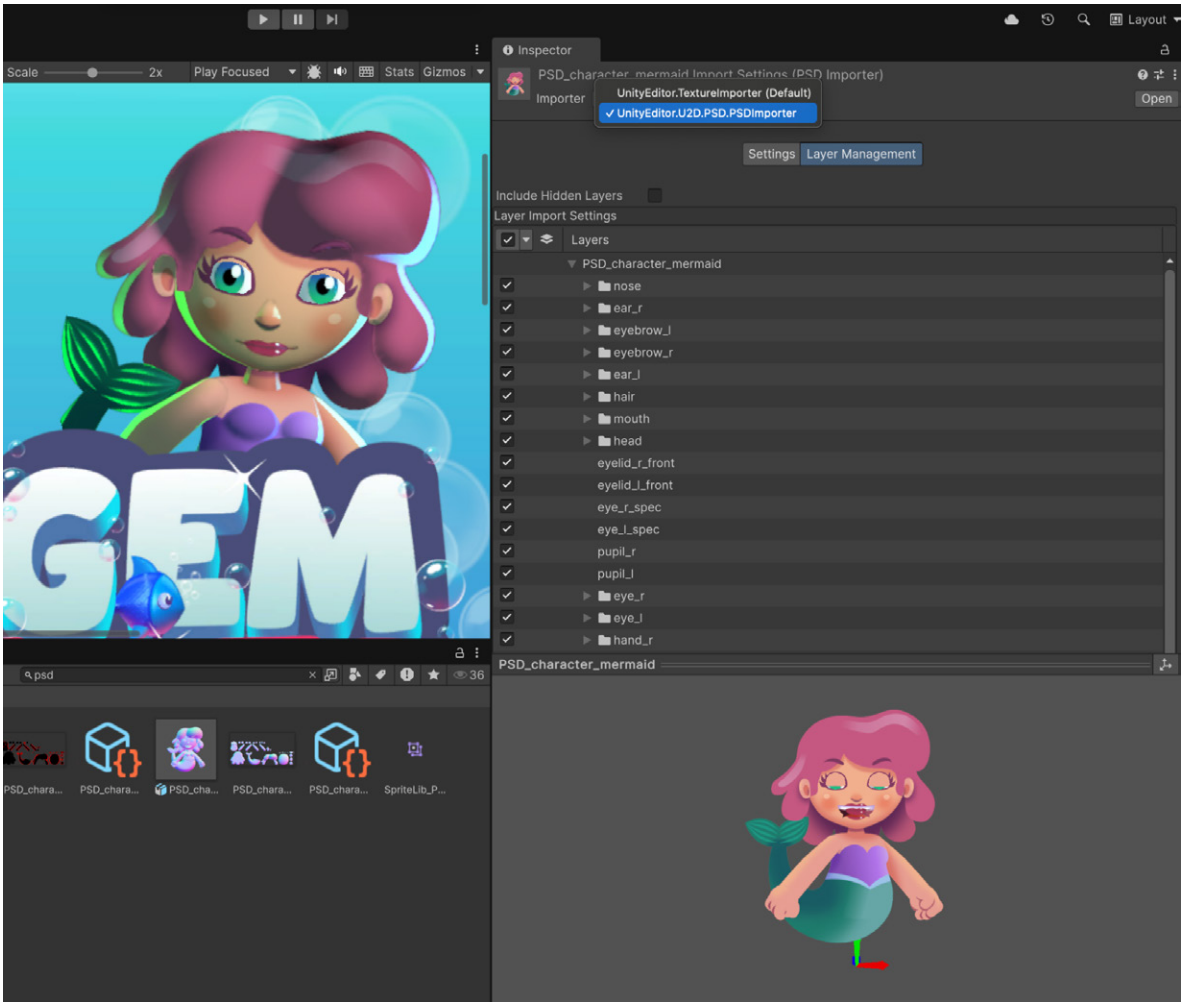
Ускорьте процесс повторного импорта с помощью PSD Importer

PSD Importer импортирует файлы Adobe Photoshop в Unity и на основе исходного файла создаёт префаб из спрайтов.

С PSD Importer можно ускорить импорт 2D-спрайтов, элементов UI и любых других ассетов, созданных в Photoshop. Раньше художникам приходилось экспортировать каждый слой файла Photoshop в отдельный файл .png для Unity, а после каждого изменения заново экспортировать все файлы .png. Теперь это не требуется: Unity поддерживает многослойный формат .psd.

Анимированных 2D-персонажей, у которых каждая конечность или другая часть находится в отдельном слое, можно импортировать в Unity для дальнейшей работы в 2D Animation. PSD Importer подготовит персонажа к последующему риггингу и анимации.

Если каждый слой PSD должен стать отдельным спрайтом, снимите флажок Character Rig. Тогда префаб создаваться не будет, а результатом станет обычный набор нарезанных спрайтов: каждый слой можно будет использовать как самостоятельный спрайт.



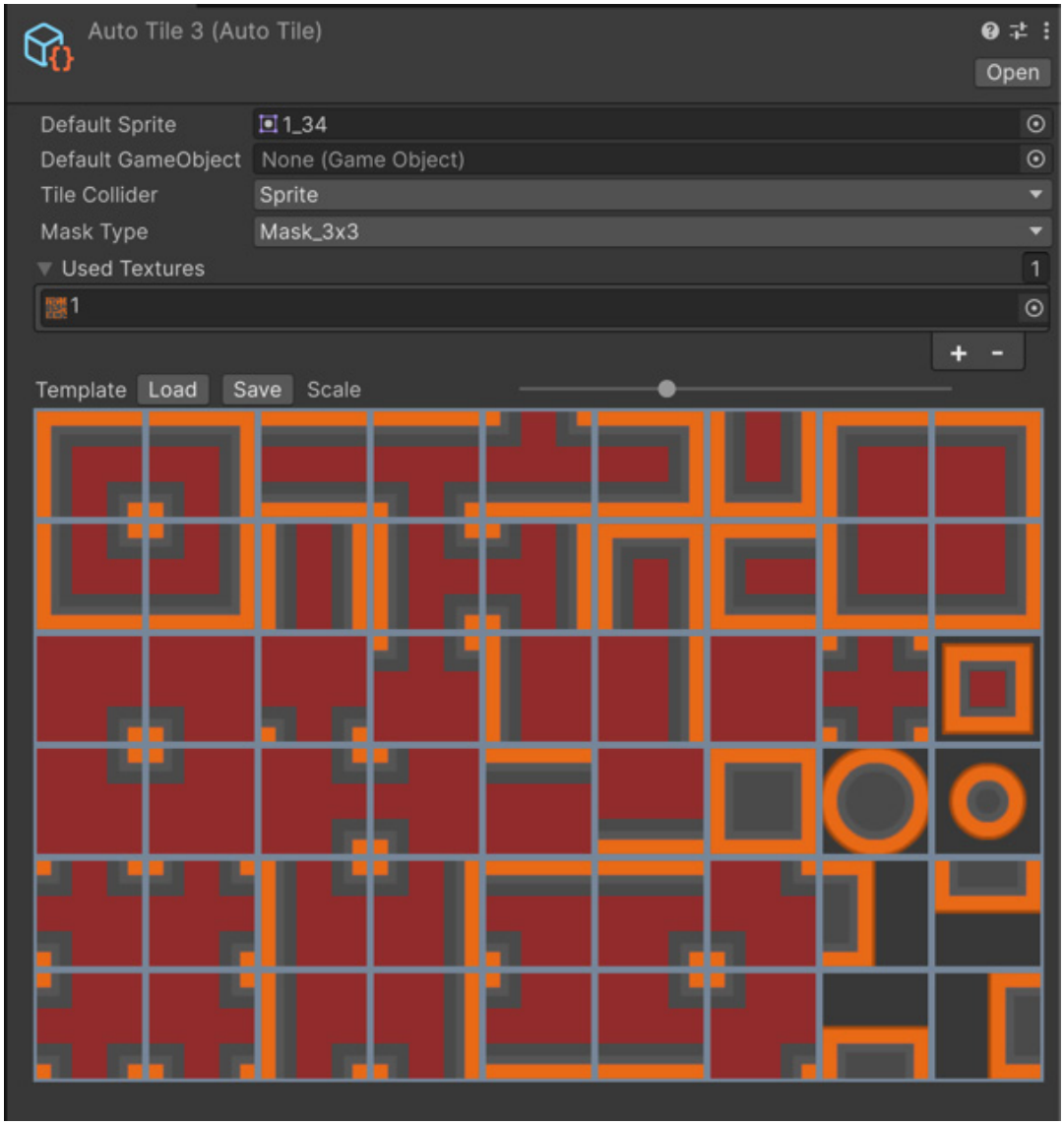
Импортируйте многослойные PSD-файлы именно через PSD Importer, а не через стандартный импортёр текстур, чтобы были доступны все возможности импортёра.

Другие полезные советы в действии смотрите в видеоруководстве «Из Photoshop в Unity с помощью PSD Importer».

Упростите работу с Rule Tile в Tilemap Editor в Unity 6.1

Система Tilemap в Unity хранит и обрабатывает ассеты Tile, предназначенные для создания 2D-уровней на сетке. Благодаря этому уровни легко создавать и итеративно дорабатывать прямо в Unity.

При работе с тайлами бывает сложно создать ассеты всех соседних вариантов тайла и назначить их ассету Rule Tile. В Unity 6.1 этот процесс упрощает новая функция AutoTile. Загрузите лист тайлов, выделите внутренние области и сохраните его: Unity создаст ассет тайла, который можно использовать в палитре тайлов.

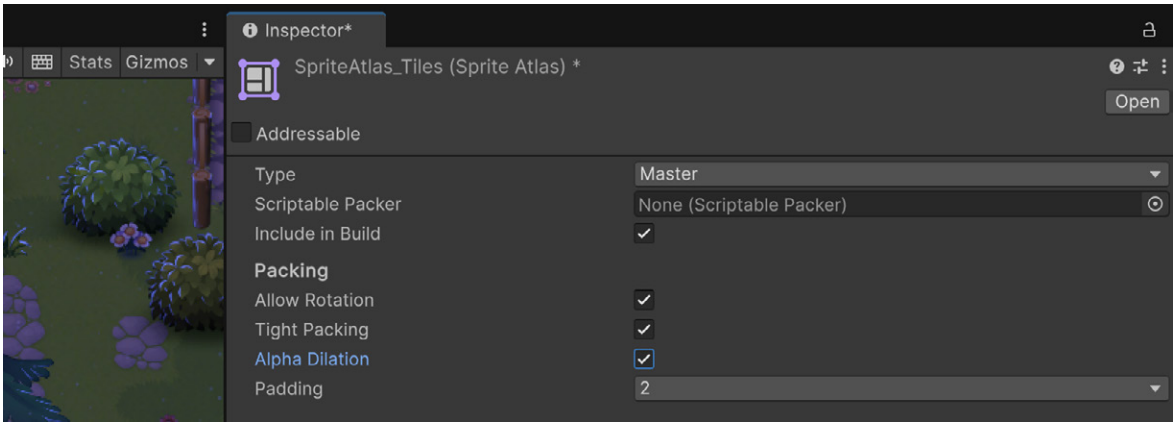


Пример AutoTile

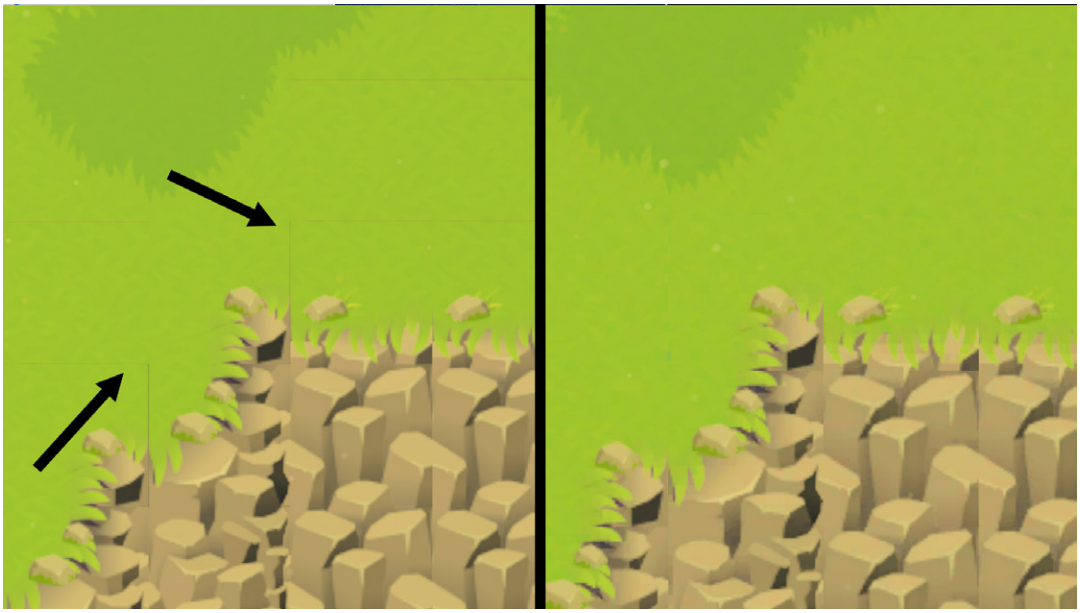
Избегайте затекания текстур и зазоров между тайлами

После всей работы над графикой и тайлмапами важно не допустить просачивания текстуры и появления небольших зазоров между тайлами из-за интерполяции и сглаживания краёв. Для игр с пиксельной графикой, где спрайты не сглаживаются, эта проблема нехарактерна.

Если вы не используете лист тайлов, рекомендуем упаковать спрайты в Sprite Atlas, чтобы сгладить стыки, и включить внутреннюю сортировку в TilemapRenderer. Это улучшит организацию проекта, производительность и творческий контроль. Настройки по умолчанию подходят большинству спрайтов, но для тайлмапов может потребоваться дополнительная настройка. Чтобы тайлы сохраняли чёткие края, при упаковке отключите Allow Rotation и используйте Alpha Dilation.



Параметр Alpha Dilation в Sprite Atlas



На левом изображении заметно просачивание по краям тайлов; проблему устранили, скорректировав настройки Sprite Atlas.

Используйте систему 2D Inverse Kinematics для создания естественных движений

Unity включает комплексное решение для скелетной анимации под названием 2D Animation. Для управления движением рига используйте систему Unity 2D Inverse Kinematics (IK). С помощью IK можно создавать естественные и динамичные движения, например сгибание коленей или вытягивание руки, без ручной настройки каждой кости.

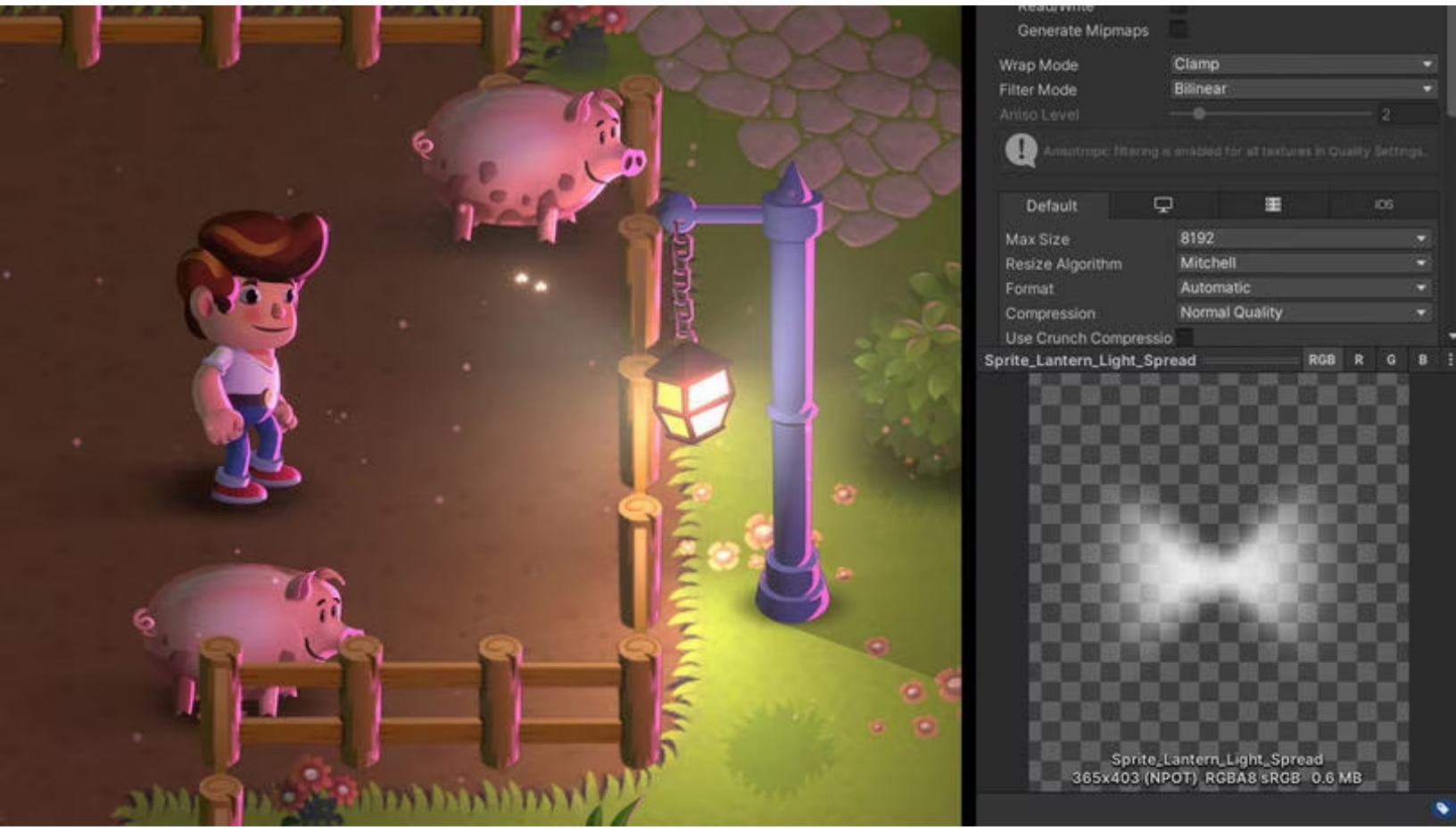
Использование 2D IK при анимации главного персонажа в демонстрационном проекте Happy Harvest

Тщательно проектируйте иерархии костей в Sprite Editor, чтобы обеспечить корректное управление. Например, сделайте кость кисти дочерней по отношению к кости руки. Проверьте иерархию на простых анимациях и убедитесь, что изменения кости верхнего уровня правильно влияют на её дочерние кости.

Используйте Animation Blend Trees для плавных переходов между анимациями ходьбы, бега и покоя в зависимости от таких переменных, как скорость или направление ввода. Это позволяет добиться естественных анимаций без резких переключений между состояниями.

Имитируйте профили IES с помощью 2D-источников света

В 3D-приложениях, например в HDRP, затухание света часто имитируют с помощью профилей IES. Система 2D-освещения Unity очень гибкая и предлагает удобные параметры настройки, включая цвет, интенсивность, затухание и эффекты смешивания. Если требуется точнее настроить характер затухания источника света, выберите для него тип Sprite и используйте созданный вами световой эффект.

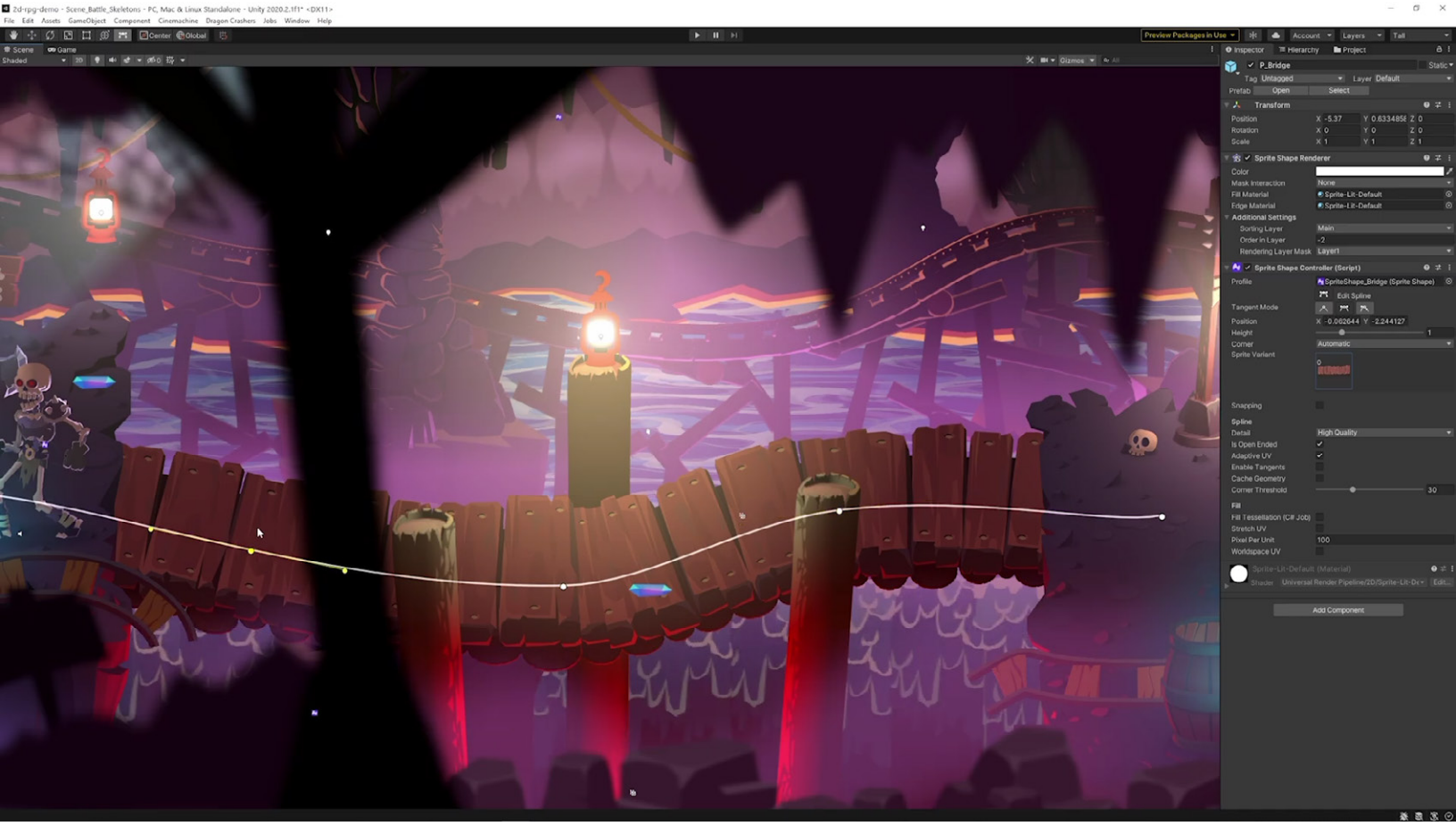


Спрайт с ореолом вокруг подвесной лампы в демонстрационном проекте Happy Harvest

Дополнительные советы по 2D-освещению можно найти в этой статье Мартина Райнмана из Odd Bug Studio, а приёмы работы с 2D-светом и тенями в Universal Render Pipeline - на этой странице.

Создавайте детализированные 2D-окружения произвольной формы с помощью 2D Sprite Shape

2D Sprite Shape позволяет создавать детализированные 2D-окружения произвольной формы с помощью наглядного и интуитивно понятного рабочего процесса. Инструмент укладывает спрайты вдоль контура фигуры, автоматически деформируя и заменяя их в зависимости от угла контура.



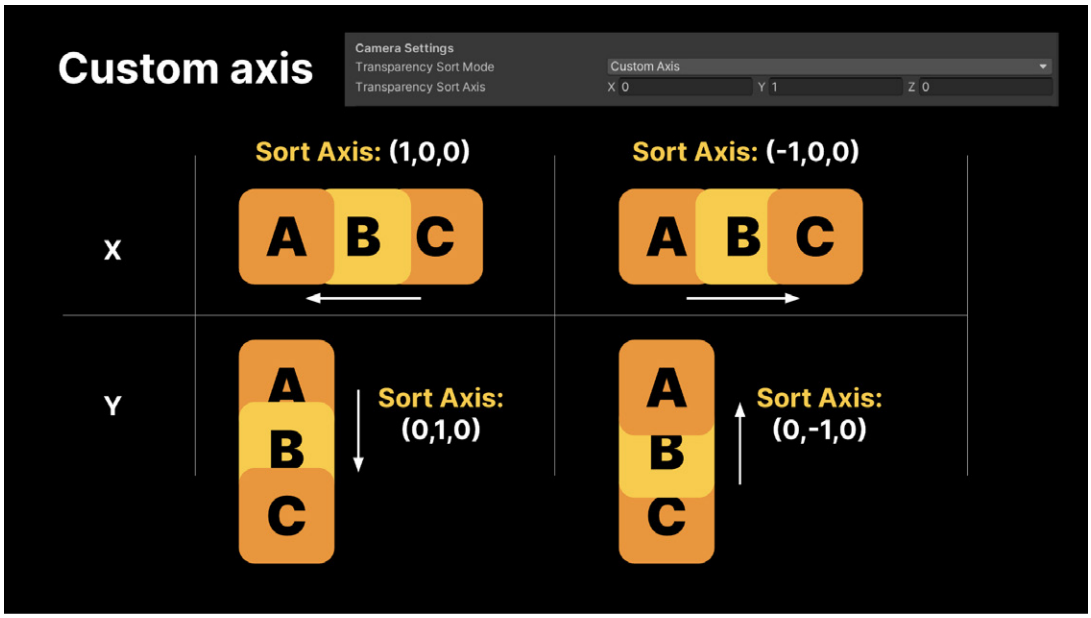
Пример использования 2D Sprite Shape в демонстрационном 2D-проекте Dragon Crashers

Sprite Shape также автоматически создает компонент Polygon Collider 2D, если вы хотите использовать контуры или фигуры в системе Physics 2D.

Пользовательская ось сортировки спрайтов

Сортируйте спрайты в выбранном направлении. Это полезно, когда несколько спрайтов находятся на одном слое, имеют одинаковый порядок сортировки, но должны отрисовываться в определенной последовательности. Например, представьте карточную игру, где карты немного перекрываются: при этом карта в нижней части экрана должна отрисовываться поверх остальных.

В URP откройте ассет 2D Renderer и выберите General > Transparency Sort Mode > Custom Axis. Например, задайте для Transparency Sort Axis значение (0, 1, 0), чтобы сортировать объекты вдоль оси Y сверху вниз.



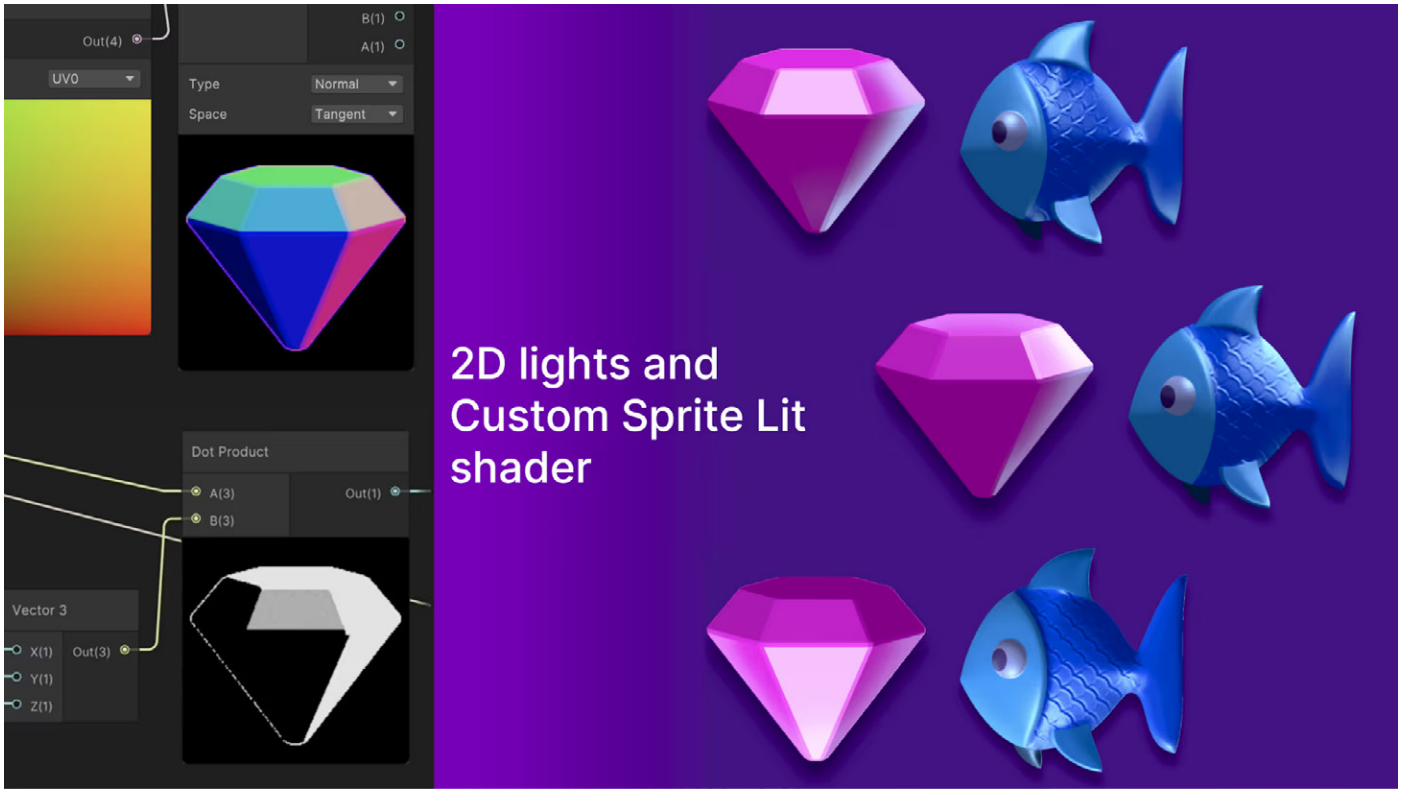
Пример настройки Transparency Sort Mode и Transparency Sort Axis; дополнительные советы см. в этом видеоруководстве (оно создано для более ранней версии Unity, но по-прежнему актуально)

Создавайте освещение с шейдерами Sprite Custom Lit

Если вам нужны эффекты освещения, не зависящие от глобального освещения сцены, рассмотрите возможность использования шейдера Sprite Custom Lit.

Это один из приемов, использованных для создания визуальных эффектов в демонстрационном проекте Gem Hunter Match. Этот шейдер заменяет освещение сцены, позволяя команде изменять данные текстуры 2D-освещения и управлять освещением каждого игрового элемента. В результате спрайты получают выразительное освещение, например переливающийся блик, движущийся по игровым элементам.

Данные о положении источника света передаются в шейдер, поэтому отдельные объекты источников света в сцене не нужны, а сама сцена остаётся чище. Инкапсулированное в шейдере освещение каждого объекта упрощает независимую настройку и массовое редактирование, а там, где возможна пакетная отрисовка, повышает производительность.

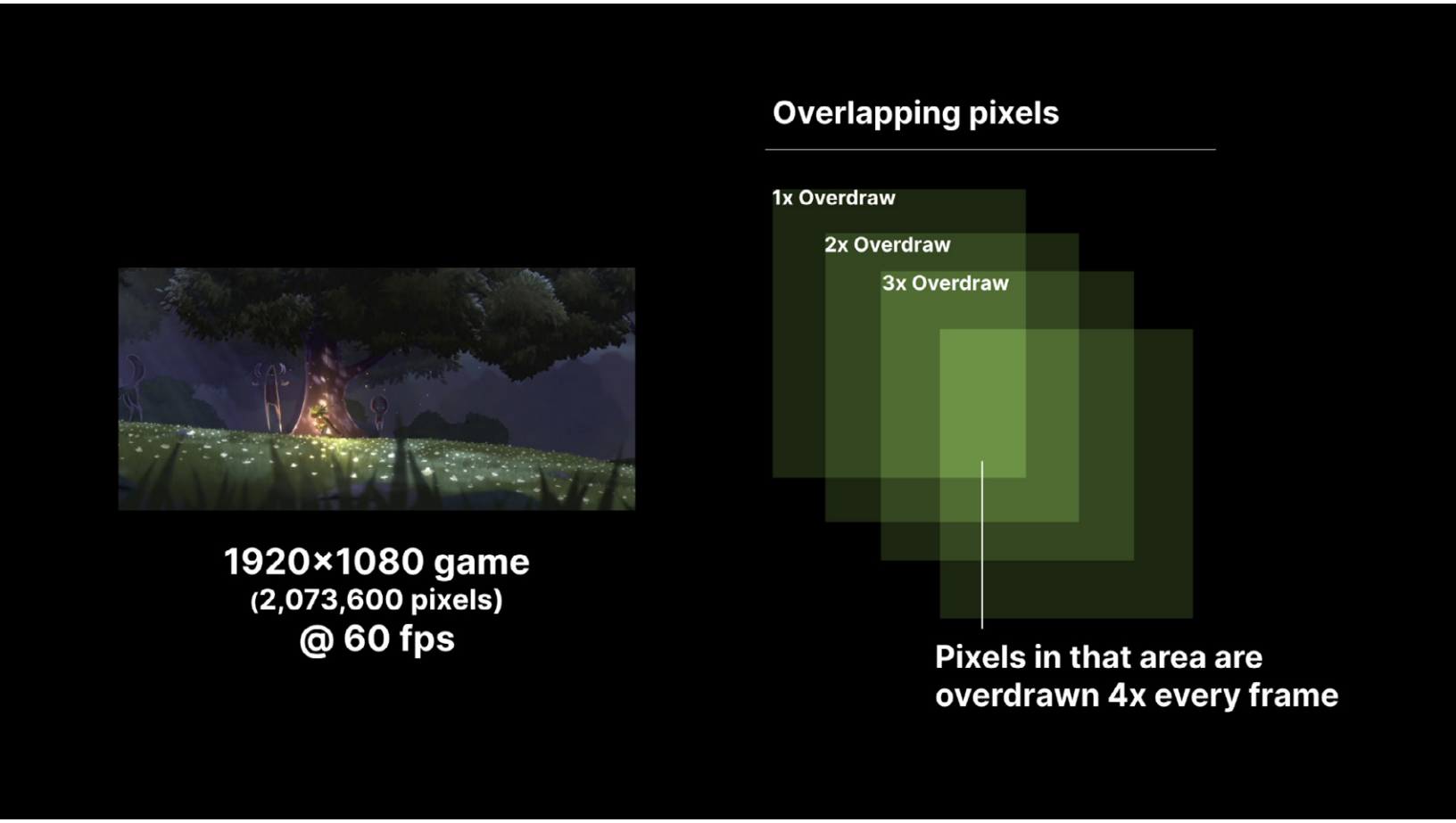


Шейдер Sprite Custom Lit в Gem Hunter Match

Узнайте из статьи в блоге «Сокровищница освещения и визуальных эффектов в новом match-3-примере Gem Hunter Match», как команда с помощью шейдера Custom Sprite Lit получила полный творческий контроль над освещением каждого 2D-спрайта.

Сократите избыточную отрисовку прозрачных пикселей

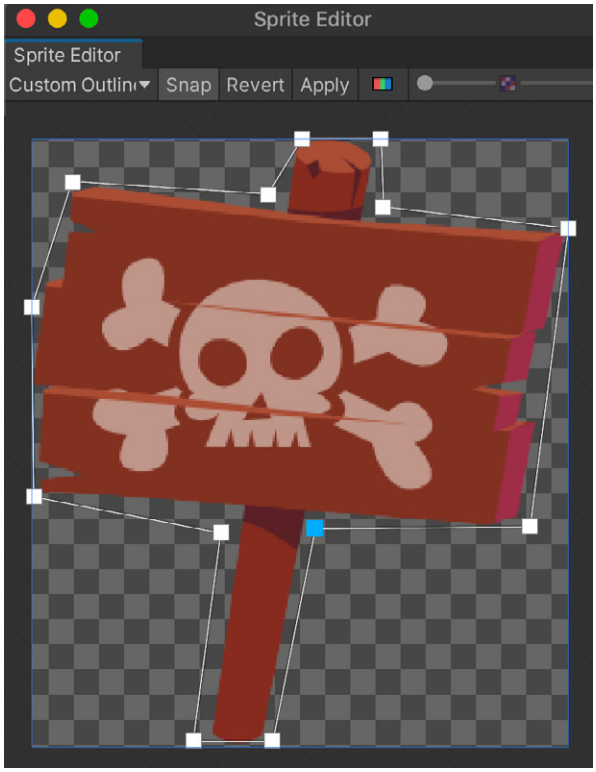
Чтобы повысить производительность, избегайте многократной отрисовки одних и тех же пикселей. В Import Settings установите для Mesh Type значение Tight (значение по умолчанию). По возможности объединяйте перекрывающуюся графику в один спрайт и отключайте спрайты в фоновых слоях, если они не используются в игре. Это уменьшит область избыточной отрисовки и вероятность перекрытия с соседними спрайтами.



Сокращение избыточной отрисовки между спрайтами

Сокращайте неиспользуемые области в Sprite Editor

С помощью 2D Sprite Editor можно также задать пользовательский контур для каждого спрайта. Это позволяет свести к минимуму неиспользуемые области при упаковке спрайтов и избежать избыточной отрисовки.

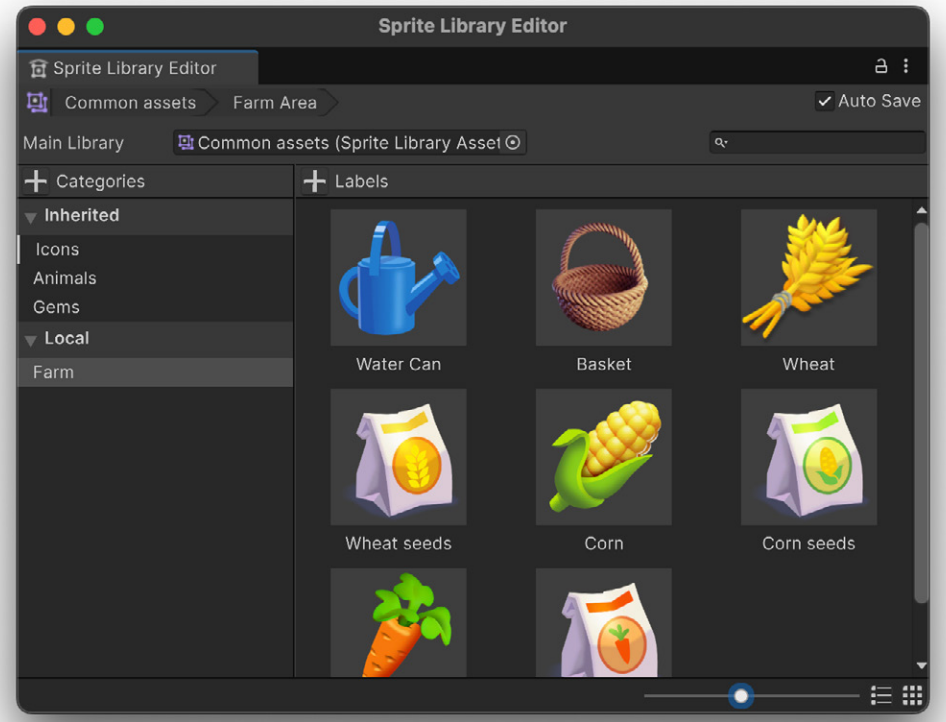


Пользовательский контур в Sprite Editor

Организуйте спрайты и анимации проекта с помощью Sprite Libraries

Используйте Sprite Libraries, чтобы удобно организовать спрайты для проектирования уровней и составных персонажей.

Sprite Libraries отличаются от Sprite Atlases, предназначенных для группировки спрайтов с целью повышения производительности отрисовки. Поскольку Sprite Libraries, Sprite Atlases и Addressables совместимы, их можно использовать вместе, чтобы поддерживать порядок в крупных проектах.



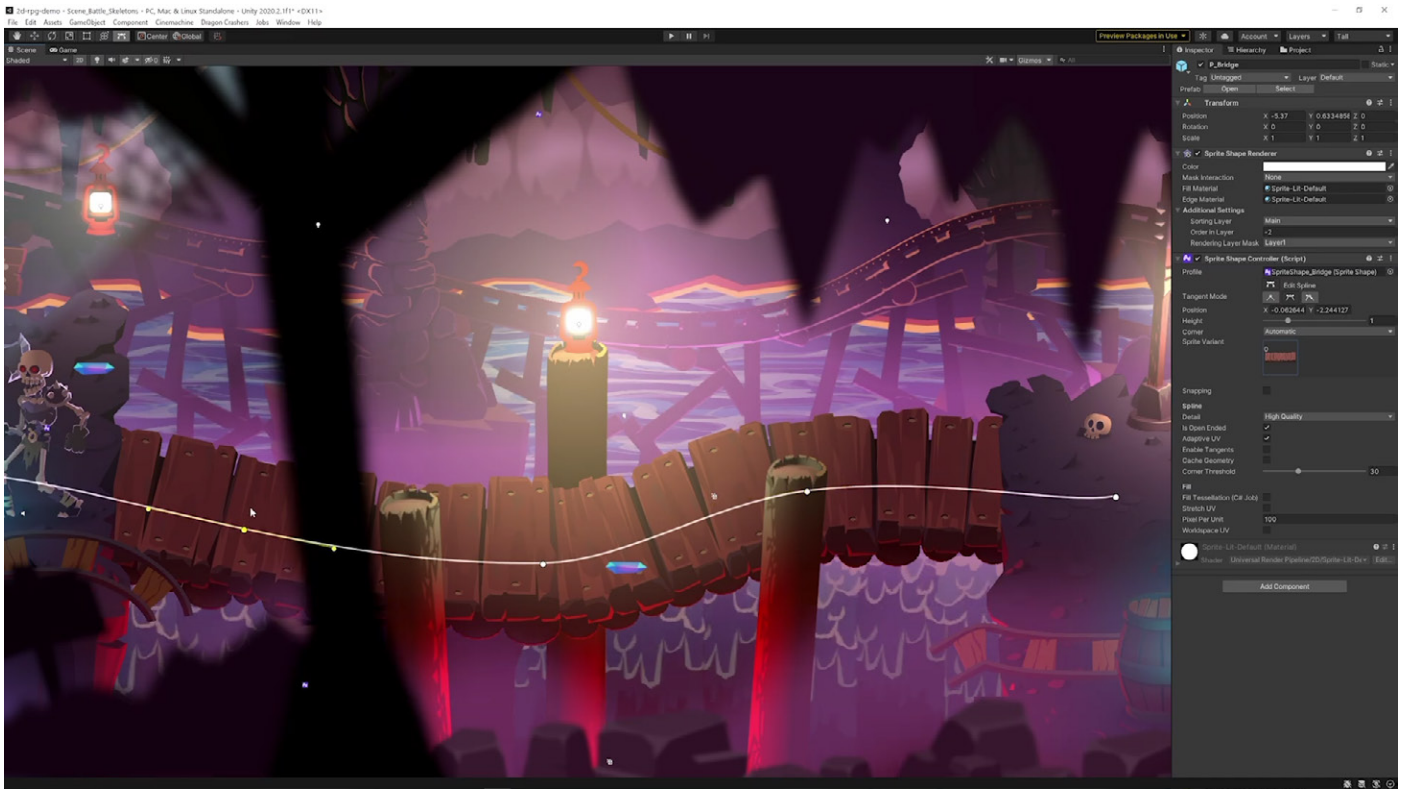
Окно Sprite Library Editor

Изменение контрольных точек Sprite Shape

Sprite Shape помогает проектировать естественно выглядящий 2D-ландшафт: холмы, динамически изменяемые дороги, тропы, гоночные трассы и водоемы, а также декоративные элементы.

С его помощью можно создавать и игровые механики: объедините Sprite Shape с системой Unity 2D Physics, чтобы создавать интерактивные области, которые способны адаптироваться и изменять форму в реальном времени, например разрушаемый ландшафт или деформируемые платформы.

Контрольные точки Sprite Shape можно изменять, а API предоставляет к ним доступ как во время выполнения, так и в редакторе. Это влияет на производительность, но при продуманной интеграции может дать игре ощутимое преимущество.



Изменение контрольных точек Sprite Shape в демонстрационном проекте Dragon Crashers - 2D URP

Удобно заменяйте спрайты через контекстное меню

Sprite Resolver - компонент, который работает вместе с системой 2D Sprite Library и позволяет динамически переключаться между различными вариантами спрайта во время выполнения.

Он позволяет менять спрайт объекта GameObject, не назначая его вручную в окне Inspector, что расширяет возможности управления спрайтами в проекте и делает его более гибким.

Вот несколько наиболее удачных сценариев использования Sprite Resolver:

- Системы настройки внешнего вида персонажей

- Динамическая смена выражений лица или настроения NPC и персонажей

- Смена спрайтов окружения в зависимости от времени года или внутриигровых событий

- Изменение внешнего вида объектов при улучшении или ухудшении их состояния

- Более эффективная и упорядоченная замена спрайтов в анимациях

В Unity 6 компоненты Sprite Resolver можно использовать как в окне Inspector, так и в компоновке редактора.

Компонент Sprite Resolver в Sprite Library

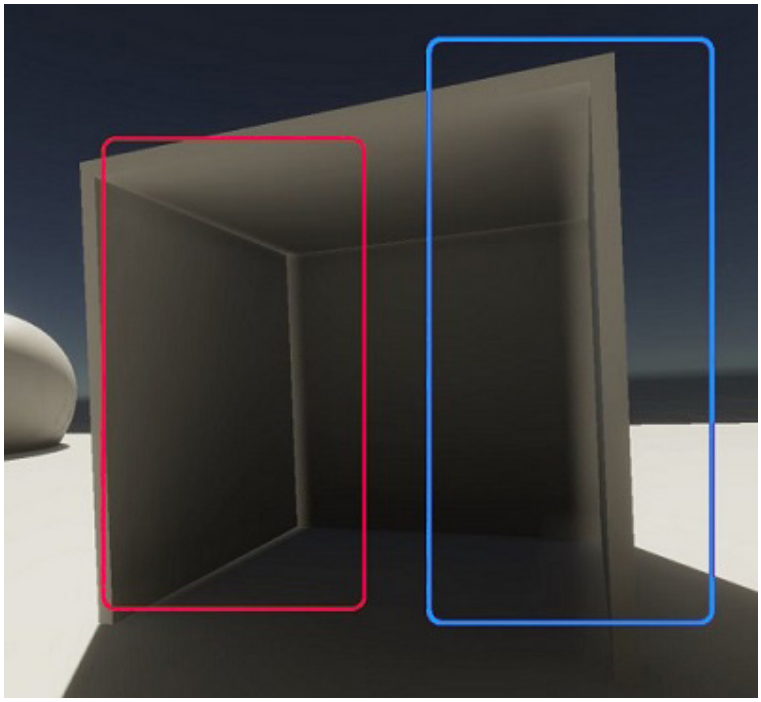
Графика и художественные ассеты

Утечки света при использовании световых зондов

Утечки света часто возникают, когда геометрия получает освещение от светового зонда, не находящегося в прямой видимости, например расположенного по другую сторону стены.

Это сложная тема, но устранить утечки света могут помочь следующие приемы:

- Сделайте стены толще
- Добавьте в сцену Adaptive Probe Volumes Options Override
- Включите Rendering Layers
- Настройте свойства Baking Set
- Используйте Probe Adjustment Volume

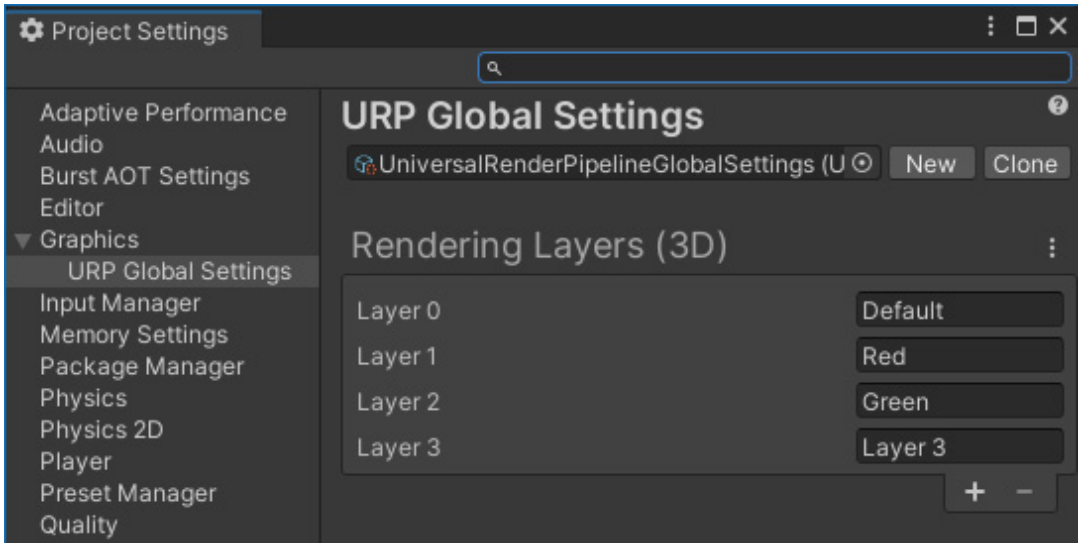


Пример утечки света

Настройте отдельные источники света для определённых объектов GameObject с помощью Rendering Layers

Функция Rendering Layers позволяет настроить источники света так, чтобы они воздействовали только на определенные объекты GameObject.

С помощью свойства Custom Shadow Layers можно настроить определенные объекты GameObject так, чтобы они отбрасывали тени только от указанных источников света, даже если сами эти источники на объекты GameObject не воздействуют.

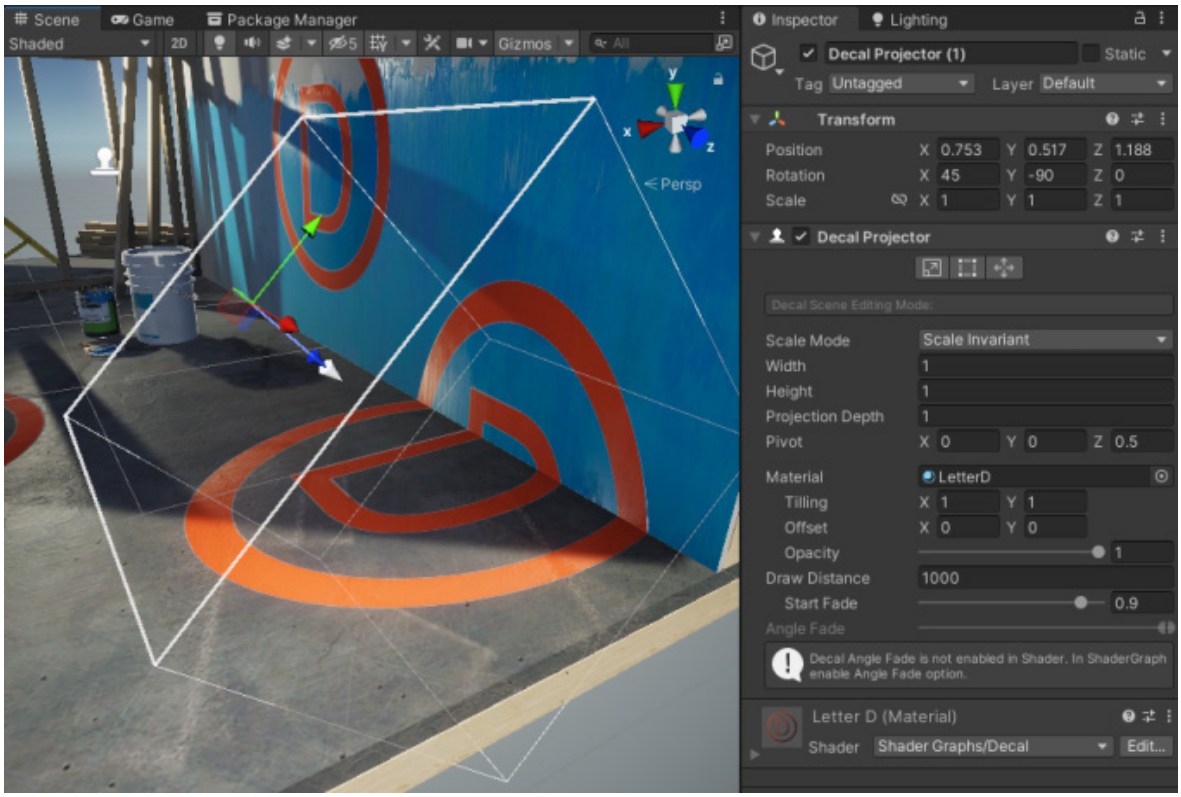


Rendering Layers в глобальных настройках URP

Добавляйте детали на меши с помощью Decal Projector

Компонент Decal Projector - отличный способ добавить детали на меш. С его помощью можно создавать такие элементы, как следы от пуль и шагов, надписи, трещины и многое другое.

Благодаря проекционному принципу декали повторяют неровную или изогнутую поверхность. Чтобы использовать Decal Projector в URP, найдите ассет Renderer Data и добавьте Decal Renderer Feature.



Decal Projector в окне Scene

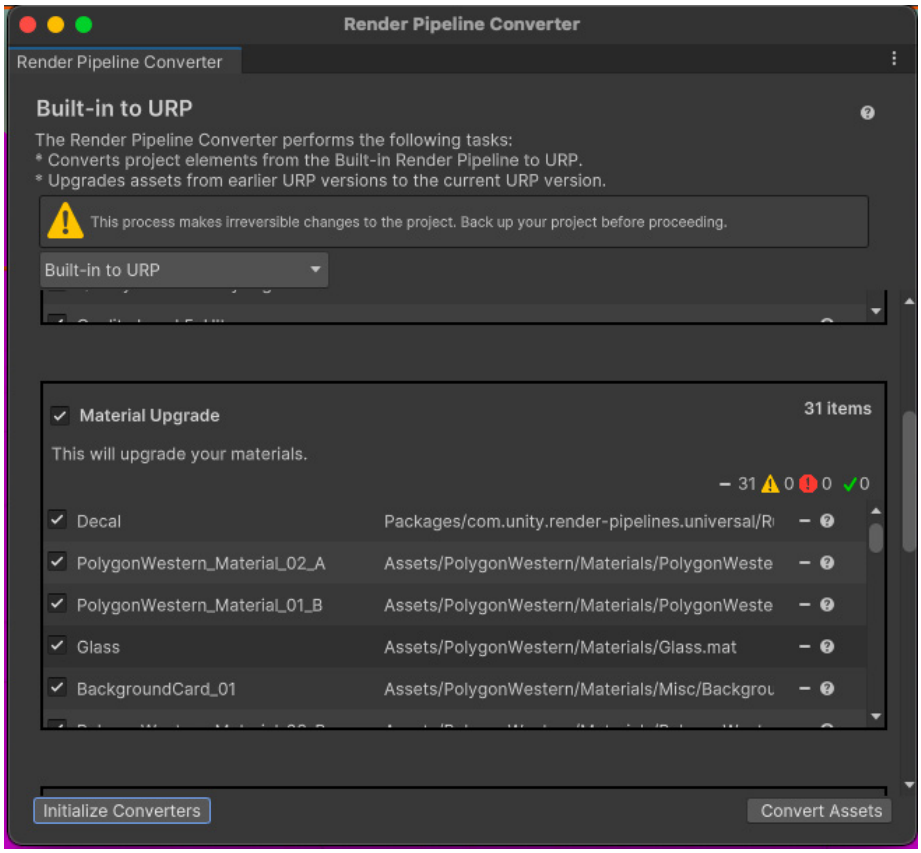
Перенесите пользовательские шейдеры из Built-In Render Pipeline в URP

Сверьтесь с этой таблицей в документации URP: в ней показано соответствие каждого шейдера URP его аналогу из Built-In Render Pipeline.

Выберите один или несколько конвертеров, затем нажмите Initialize Converters или Initialize And Convert. В обоих случаях проект будет просканирован, а ассеты, требующие преобразования, появятся на панелях соответствующих конвертеров.

Если выбрать Initialize Converters, можно ограничить набор преобразуемых элементов, сняв флажки рядом с ненужными. Затем нажмите Convert Assets, чтобы начать преобразование. При выборе Initialize And Convert преобразование начнется автоматически

после инициализации конвертеров. После завершения процесса Unity может предложить повторно открыть сцену, которая активна в редакторе.

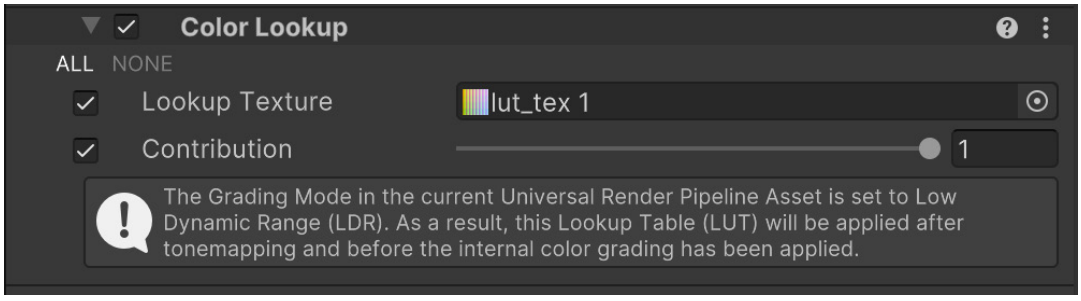


Render Pipeline Converter

Выполняйте цветокоррекцию с помощью LUT-текстур

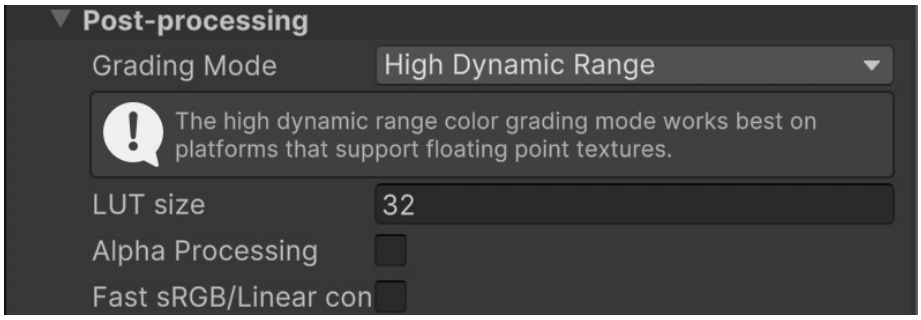
Эффект Color Grading изменяет или корректирует цвет и яркость итогового изображения, создаваемого Unity. С его помощью можно изменить визуальный стиль и атмосферу проекта.

Чтобы управлять действием Color Grading, используйте параметры Lookup Texture и Contribution в эффекте постобработки Color Lookup, который необходимо добавить в настройки Volume.

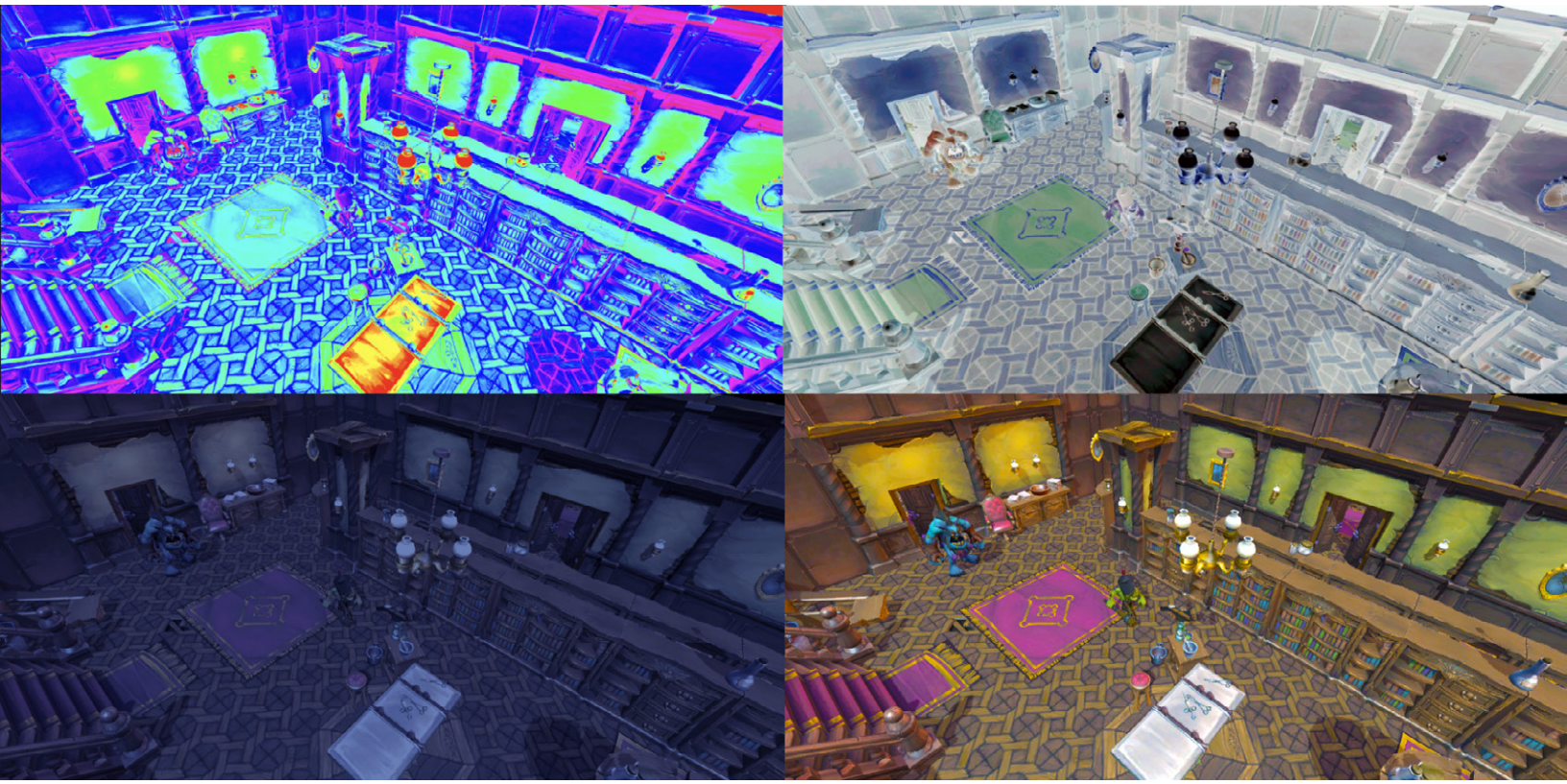


Параметры Lookup Texture и Contribution в Color Lookup

Поддерживается смешивание между несколькими LUT-текстурами низкого динамического диапазона (LDR), но оно работает корректно, только если все текстуры имеют одинаковый размер. Поэтому рекомендуется использовать во всем проекте LUT одного размера: 256x16 или 1024x32.



Настройка размера LUT в настройках ассета URP



Использование различных LUT-текстур

Создавайте реалистичные оптические эффекты и стилизованный вид с помощью бликов объектива

Блик объектива - это артефакт, возникающий, когда яркий свет попадает в объектив камеры. Он может выглядеть как один яркий засвет или как множество цветных многоугольных бликов, форма которых соответствует диафрагме камеры.

Чтобы познакомиться с бликами объектива, установите примеры через Package Manager. В проект будет добавлен набор готовых ассетов Flare для создания эффектов бликов, а также тестовая сцена, где можно изучить эти эффекты и научиться создавать собственные.

Примеры бликов объектива содержат наборы настроек - от реалистичных оптических эффектов до более стилизованных решений; здесь показан пример со сценой из HDRP Samples.

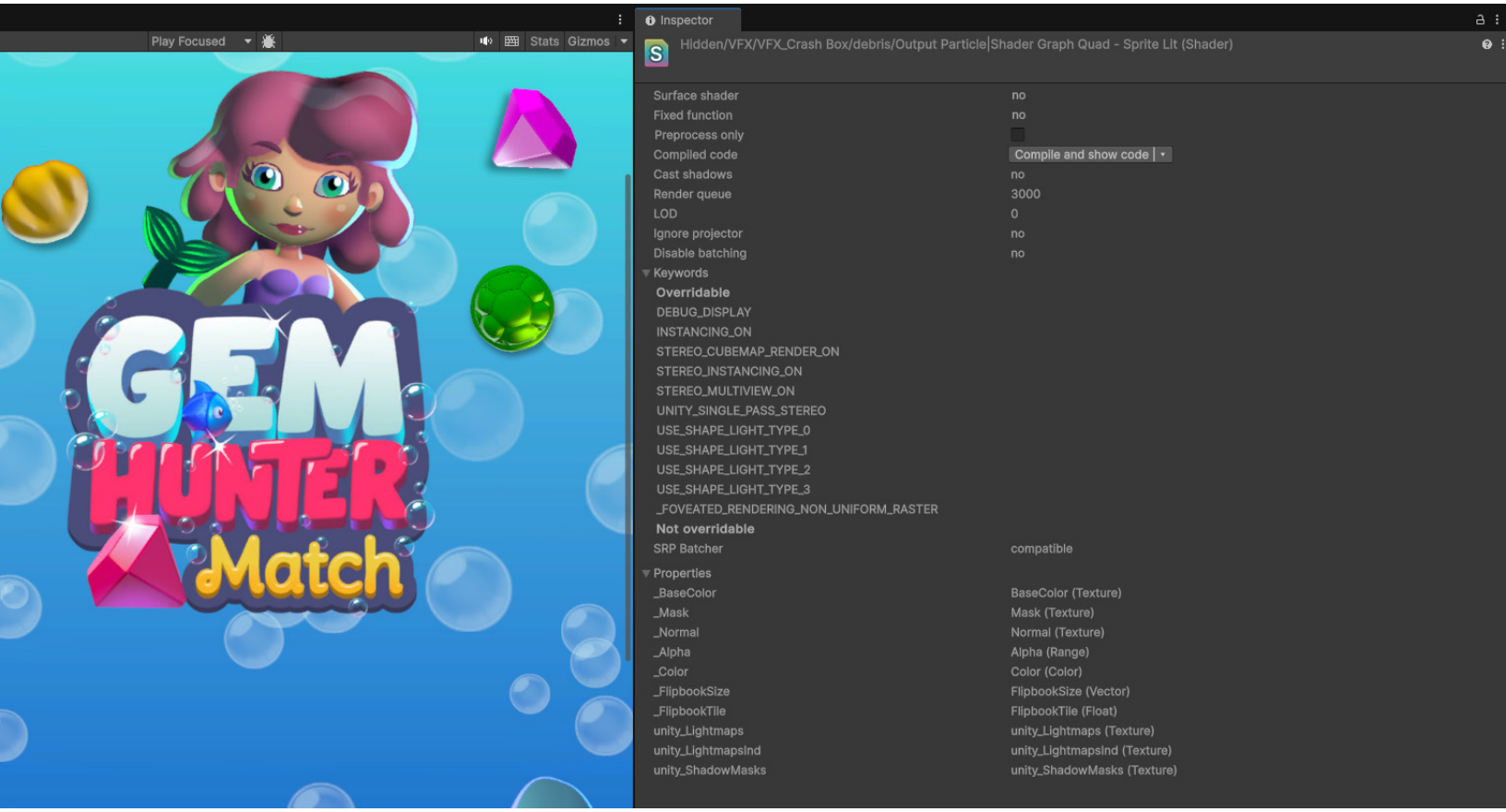
Блики объектива состоят из элементов Lens Flare. Каждый элемент представляет отдельный артефакт, создаваемый бликом. Формой элемента может быть многоугольник, круг или пользовательское изображение.

Параметры элемента позволяют настраивать Color, положение Transform и деформацию, а также расположение, цвет и масштаб элемента в многоэлементном эффекте Lens Flare. Если эффект связан с источником света, его элементы могут использовать оттенок этого источника. Благодаря этому один и тот же ассет Flare можно применять с разными источниками света.

Подробнее о принципах работы бликов объектива рассказывается в этом докладе с конференции SIGGRAPH.

Управляйте вариантами шейдеров

Чтобы сократить размер и время сборки, тщательно управляйте вариантами шейдеров. Если варианты нужны только определённым материалам, используйте #pragma shader_feature вместо #pragma multi_compile. Варианты shader_feature исключаются из сборки, если они не используются материалами и не включаются во время выполнения, тогда как варианты multi_compile включаются всегда. Проверяйте ключевые слова в окне Inspector шейдера.



Раздел Keywords в окне Universal Render Pipeline/Lit (Shader) в примере Gem Hunter Match

Создавайте варианты материалов

Material Variants позволяют создавать шаблоны и префабы материалов. На основе базового шаблона можно создавать варианты, которые наследуют общие свойства шаблонного материала и переопределяют только отличающиеся свойства. Изменения общих непереопределённых свойств шаблонного материала автоматически отражаются в его вариантах. Кроме того, отдельные свойства материала можно заблокировать, чтобы варианты не могли их переопределять.

Пример Material Variants: все варианты используют один базовый материал и отличаются только свойством цвета

В более сложной конфигурации можно создавать варианты Material Variant. Иерархия наследования материалов способствует повторному использованию, ускоряет итерации и упрощает масштабирование процесса создания материалов в проекте.

Планируйте группы ассетов для эффективной работы

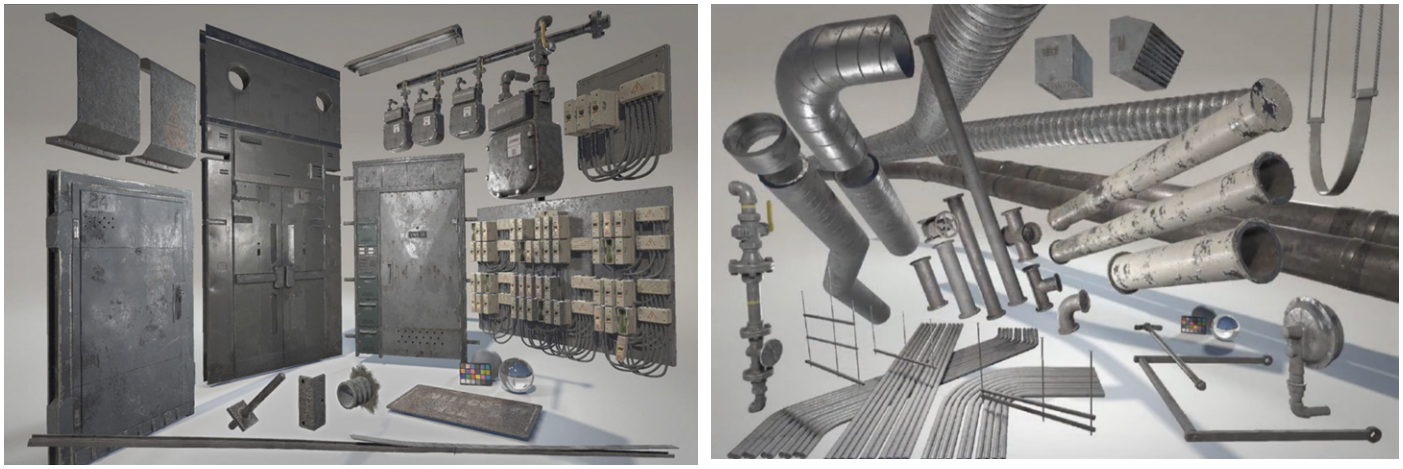
На этапе подготовки к производству команда должна согласовать организацию работы с ассетами: составить список необходимых ассетов, определить их бюджет с учётом технических требований проекта и оценить стоимость разработки.

Например, необходимые элементы уровня можно разделить на следующие группы:

- Крупные элементы. Для таких элементов нельзя создавать уникальные текстуры: следует использовать тайловые текстуры или модульные элементы. К ним относятся стены, земля, скалы, крыши зданий и т. п.

- Объекты среднего размера. Это объекты с собственными листами текстур, например бочки, ящики, камни и двери.

- Часто повторяющиеся мелкие элементы. Это предметы, которые помогают визуально связать объекты с окружением или передать масштаб: галька, листья, винты, болты и т. п.



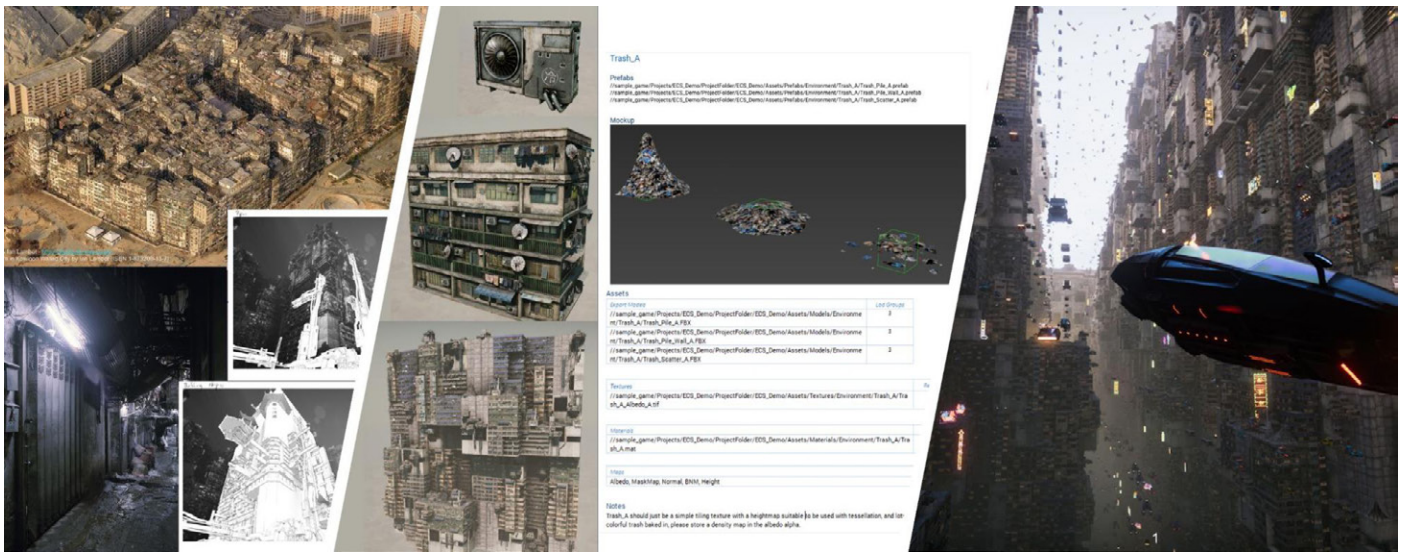
Планируйте создание ассетов с учётом их типа, размера и формы

Определите для каждого ассета бюджет времени, число полигонов и разрешение текстур, а затем расставьте приоритеты задач. После этого создавайте второстепенные ассеты, а количество критически важных ассетов сведите к необходимому минимуму.

Автоматизируйте и ускорьте конвейер импорта с помощью API AssetPostProcessor

Если в проекте используются тысячи ассетов, не настраивайте параметры каждого из них вручную в окне Inspector.

Автоматизировать проверку файлов ассетов позволяет API AssetPostProcessor. С его помощью можно подключаться к конвейеру импорта и запускать скрипты до или после импорта ассетов: достаточно добавить ассеты, и скрипты внесут необходимые изменения. Параметры можно менять в одном скрипте, который автоматически повторно применит их ко всем ассетам проекта.



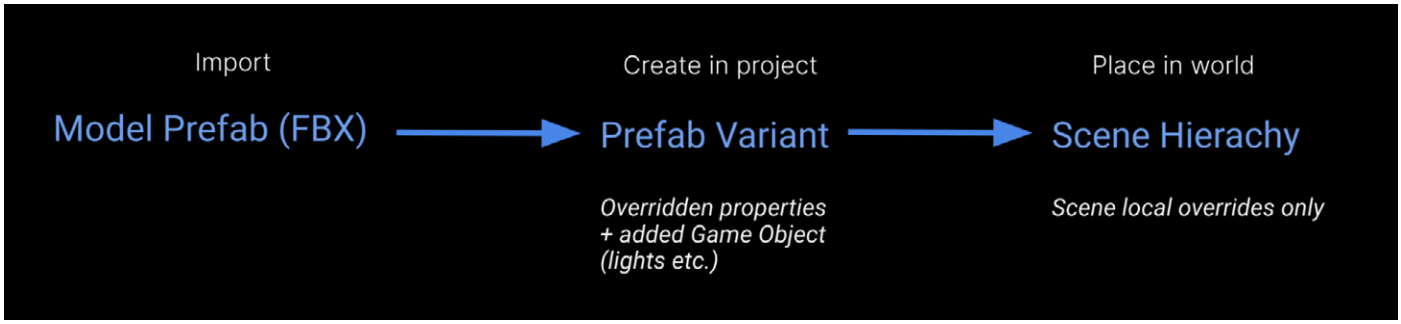
Слева направо: примеры мудборда, концепт-арта, ассетов префабов, стандартов структуры каталогов и именования с использованием AssetPostProcessor, а также итоговый вид демонстрационного проекта Unity Megacity

Эффективнее работайте в команде с вариантами префабов

Ещё один важный этап импорта ассетов - создание префабов из 3D-моделей. Не рекомендуется во время производства напрямую преобразовывать FBX-модель в префаб: изменения FBX-файла, например изменение иерархии геометрии, могут нарушить префаб. Тогда все его экземпляры в сцене придётся заменять обновлённой версией. Эту проблему решают Prefab Variants.

Prefab Variants позволяют создавать художественные ассеты независимо от сцен, поэтому хорошо подходят для прототипирования. Их рекомендуется использовать при совместных итерациях во время построения игрового мира. Они сохраняют преимущества обычных префабов, например возможность создавать варианты исходной модели, но при этом служат «слабой» ссылкой на префаб FBX-модели. Благодаря этому элементы модели можно добавлять, разделять и удалять без пересоздания префаба. Художники могут изменять FBX-модели, а Prefab Variant будет обновляться автоматически. Кроме того, Prefab Variants позволяют художникам оценивать изменения графики в контексте, не меняя сам проект.

Одновременно с этим рассмотрите возможность использовать Nested Prefabs, чтобы упростить совместную работу над сценой.



Рекомендуемый процесс работы с Prefab Variants в проектах с большим числом участников

Рекомендации по ассетам для XR и мобильной разработки

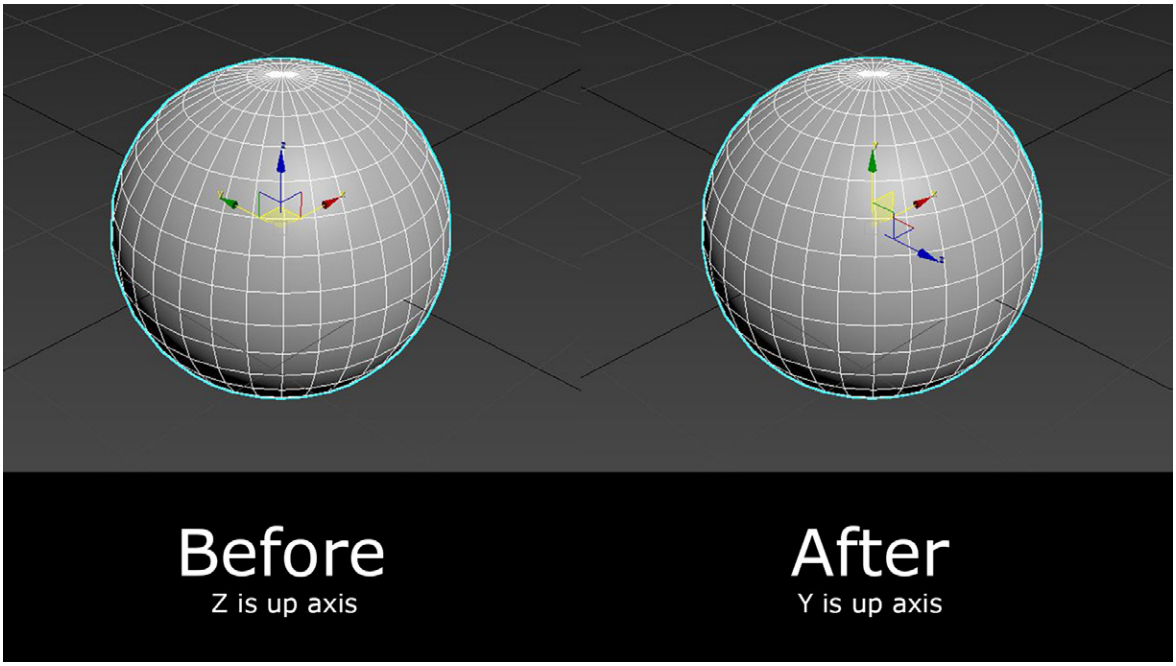
Решения, принятые на раннем этапе создания или подбора художественных ассетов для XR- или мобильной игры, помогут сэкономить время в дальнейшем. Следуйте этим рекомендациям, чтобы эффективно создавать любые художественные ассеты - от небольших объектов до персонажей.

- Начинайте с чёткого представления о том, что хотите создать. Особенно на ранних этапах концепцию можно проиллюстрировать простым рисунком от руки, чтобы быстро спланировать взаимодействия или визуальное оформление.

    - В крупной студии идеи может воплощать концепт-художник; отдельным разработчикам и небольшим командам визуализировать замысел поможет ИИ.

- Создавайте модель в программе для 3D-моделирования, например Blender, Autodesk Maya или 3ds Max. Следите за числом полигонов, поскольку оно влияет на производительность, особенно в мобильной AR. Сначала можно создать высокополигональную модель, а затем оптимизировать её.

- Настройте опорную точку модели в программе для 3D-моделирования так, чтобы её ориентация соответствовала системе координат Unity. Ниже показано, как задать опорную точку в 3ds Max, где вертикальной является ось Z.



Настройка вертикальной оси

- Ориентацию модели также можно исправить в Unity, выполнив два действия:

a. Создайте пустой объект GameObject в окне Hierarchy.

b. Сделайте модель дочерней для этого GameObject и задайте ей Position 0, 0, 0 по всем осям, чтобы точно выровнять её по центру.

В результате модель будет привязана к родительскому объекту GameObject с нейтральным Rotation 0 по всем осям. Модель также будет правильно ориентирована: ось Z направлена вперёд, а Scale по всем осям равен 1. Кроме того, в Importer можно попробовать параметр Bake Axis Conversion.

- Наложите текстуры, чтобы придать модели реалистичный или стилизованный вид. Для детализированных текстур используйте UV-развёртку. В зависимости от выбранной стилистики игре могут потребоваться PBR-текстуры.

- Шейдеры на основе физически корректного рендеринга (PBR) в Unity обычно поддерживают два основных процесса: Metallic-Roughness и Specular-Glossiness. Каждый из них отвечает за разные свойства материала. В результате 3D-сцены и объекты в играх и симуляциях на Unity выглядят более реалистично и динамично.

Заполняйте большие текстурированные области трим-листами

Использование одного трим-листа для разных мешей окружения

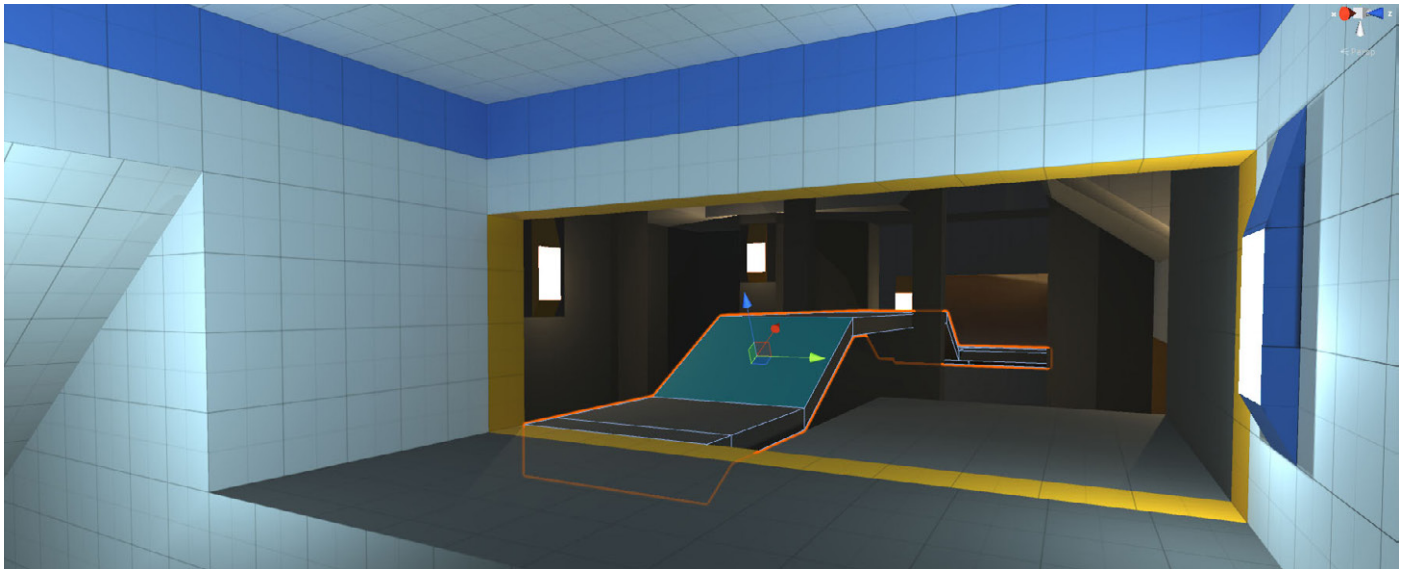
Трим-лист - это средство текстурирования, применяемое в 3D-моделировании и разработке игр. Он представляет собой одну текстуру с набором декоративных узоров и деталей: молдингов, кромок, окантовок и т. п.

Трим-листы помогают оптимизировать производительность игры: они сокращают количество отдельных текстур, но при этом позволяют создавать визуально сложное и разнообразное оформление моделей.

Экспортируйте 3D-модели с соблюдением масштаба при работе с AR и VR

Создавать прототипы и моделировать 3D-меши можно в предпочитаемом DCC-приложении либо непосредственно в Unity с помощью ProBuilder, доступного в Package Manager. В ProBuilder можно быстро собрать макет уровня, задать масштаб объектов и проверить их вид в Unity. Добившись правильного масштаба, экспортируйте модели для дальнейшей доработки, а затем верните их в Unity.

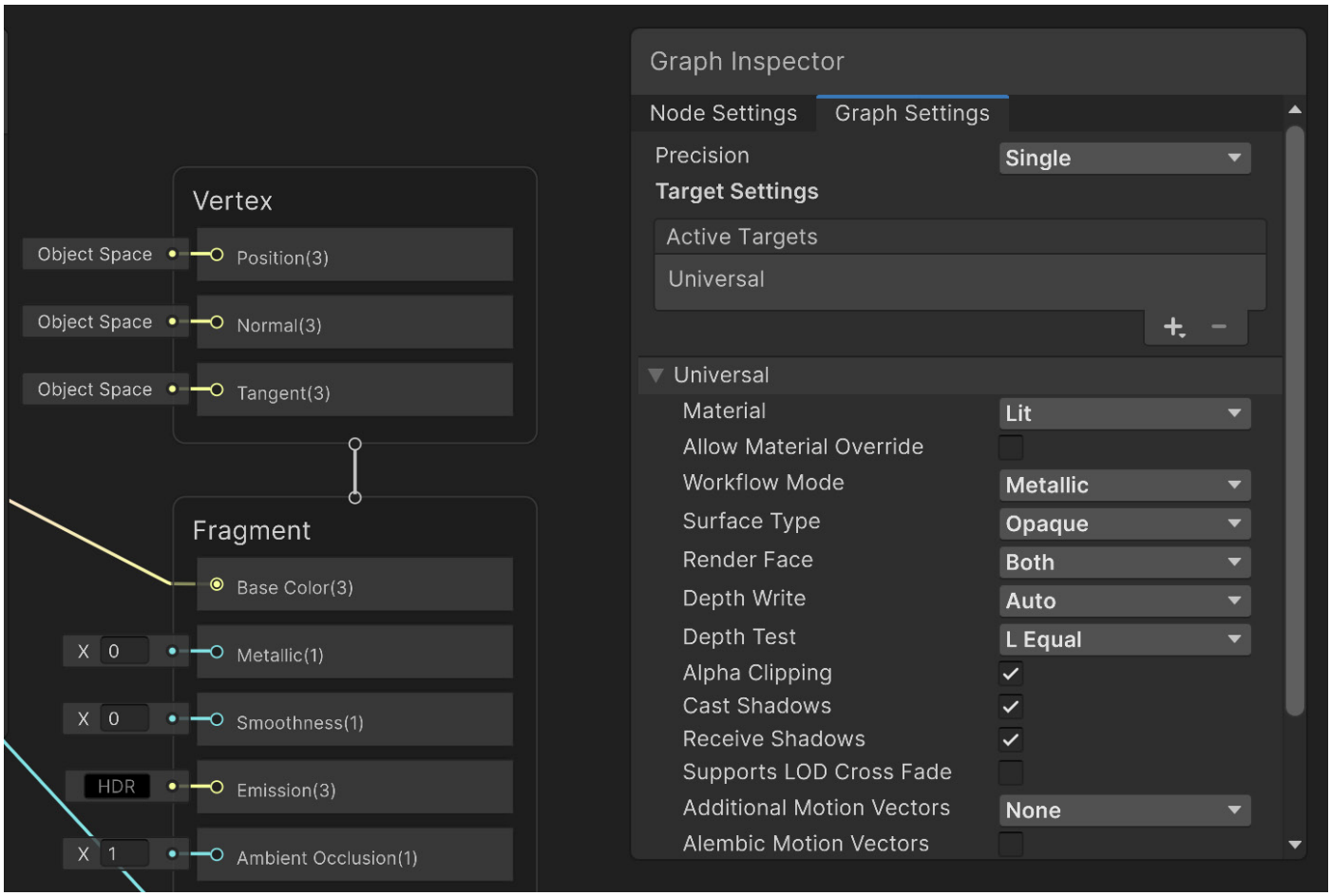
В Unity используется интуитивная шкала измерений: 1 единица точно соответствует 1 метру. Чтобы оценить масштаб импортированного ассета, добавьте в сцену куб GameObject. Куб размером 1 × 1 × 1 м служит простой и точной мерой для сравнения масштаба любого ассета. При создании окружения и объектов важно найти баланс между эстетикой и производительностью, чтобы обеспечить плавную и увлекательную работу AR- или VR-приложения.



Уровни можно прототипировать в ProBuilder, а затем экспортировать с помощью FBX Exporter в программу для 3D-моделирования и доработать там.

Shader Graph для проектов URP и HDRP

Создавая шейдеры Lit и Unlit в Shader Graph, можно упростить адаптацию ассетов для проектов как на URP, так и на HDRP. В Shader Graph конвейер рендеринга задаётся в Master Node, а остальная логика остаётся неизменной. Благодаря этому проще создавать шейдеры, работающие в обоих конвейерах.



Master Stack

UI Toolkit

Создавайте интерфейс по визуальному референсу

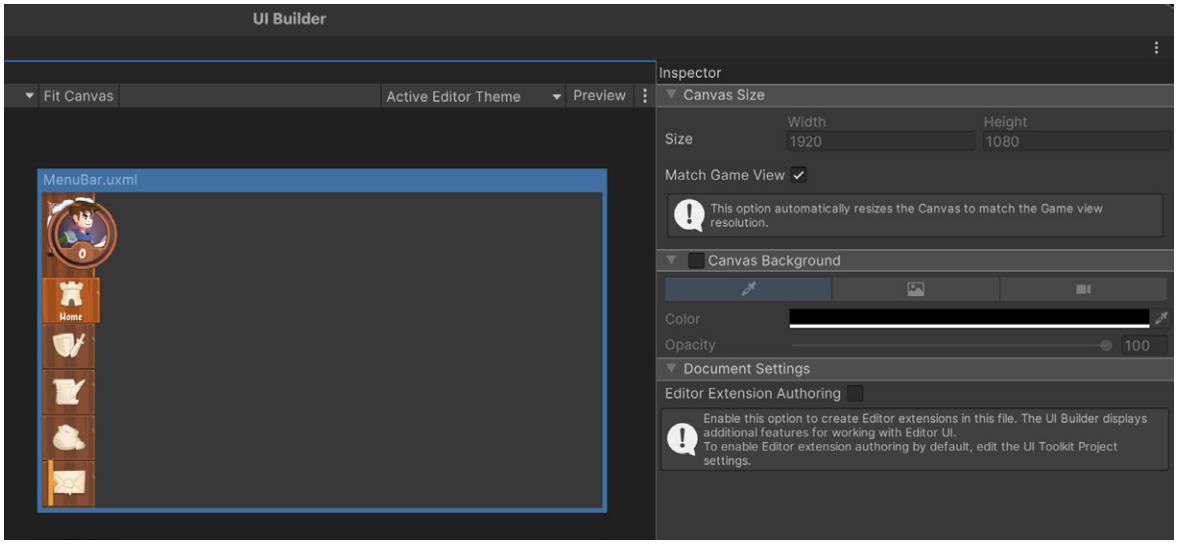
Фон Canvas помогает оценить оформление элементов поверх заданного цвета или фонового изображения. Выберите UXML-файл в панели Hierarchy, а затем задайте Canvas background, приближённый к итоговому интерфейсу, чтобы оценивать изменения стиля в контексте.

Для Canvas background доступно несколько вариантов:

- Background Color - определённый цвет или оттенок игрового окружения.

- Image - спрайт или текстура в качестве фона; удобно для воспроизведения макетов экранов и референсных изображений.

- Camera - текущий игровой процесс на фоне, позволяющий оценить UI непосредственно в контексте игры.



Canvas UXML-документа: настройка внешнего вида с помощью параметров Color и Image

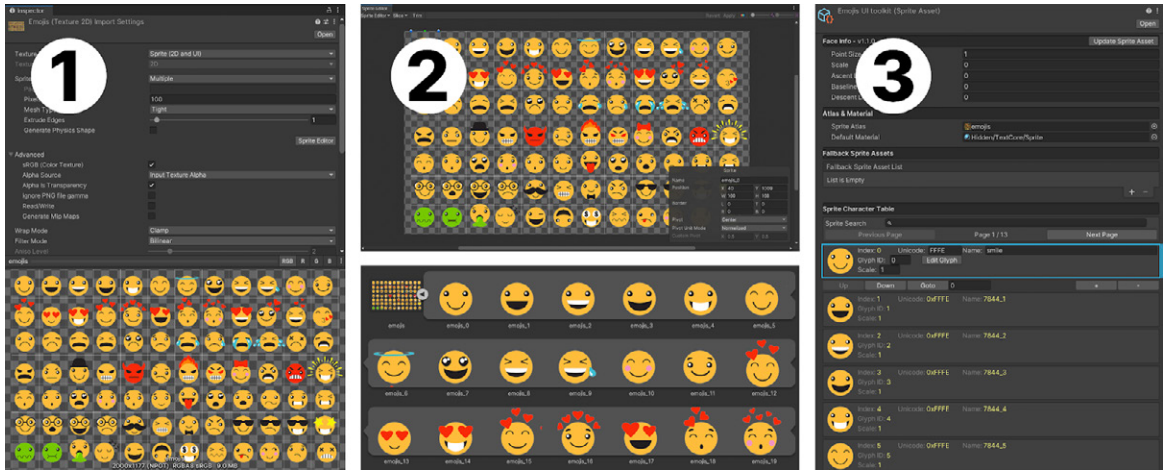
Ускорьте итерации при работе с файлами Photoshop с помощью PSD Importer

Если многослойный PSD-файл импортирован в проект с помощью PSD Importer, Unity автоматически обновляет содержащиеся в нём спрайты при каждом сохранении файла. Благодаря этому можно быстро создать черновой вариант, дорабатывать его и сразу видеть изменения в Game view. Такой подход заметно экономит время и повышает качество результата: вы оцениваете его в контексте, не заменяя файлы и не обращаясь за помощью к другому разработчику команды.

Используйте эмодзи в игре

С помощью тегов Rich Text в текст можно добавлять спрайты, например эмодзи. Для этого потребуется ассет спрайтов, аналогичный ассету Gradient.

При импорте нескольких спрайтов упакуйте их в один атлас, чтобы сократить количество вызовов отрисовки. Убедитесь, что разрешение атласа спрайтов подходит для целевой платформы.



Импортируйте спрайты, создайте Sprite Asset в папке Assets/Resources и при необходимости настройте сведения о каждом глифе.

Используйте системные эмодзи устройства

Если приложение предназначено для конкретной платформы выполнения, например iOS или Android, вместо добавления исходного шрифта в проект можно использовать встроенный системный шрифт эмодзи. Это экономит память и избавляет от необходимости включать в приложение большую коллекцию эмодзи.

Чтобы использовать в проекте эмодзи операционной системы, выполните следующие действия:

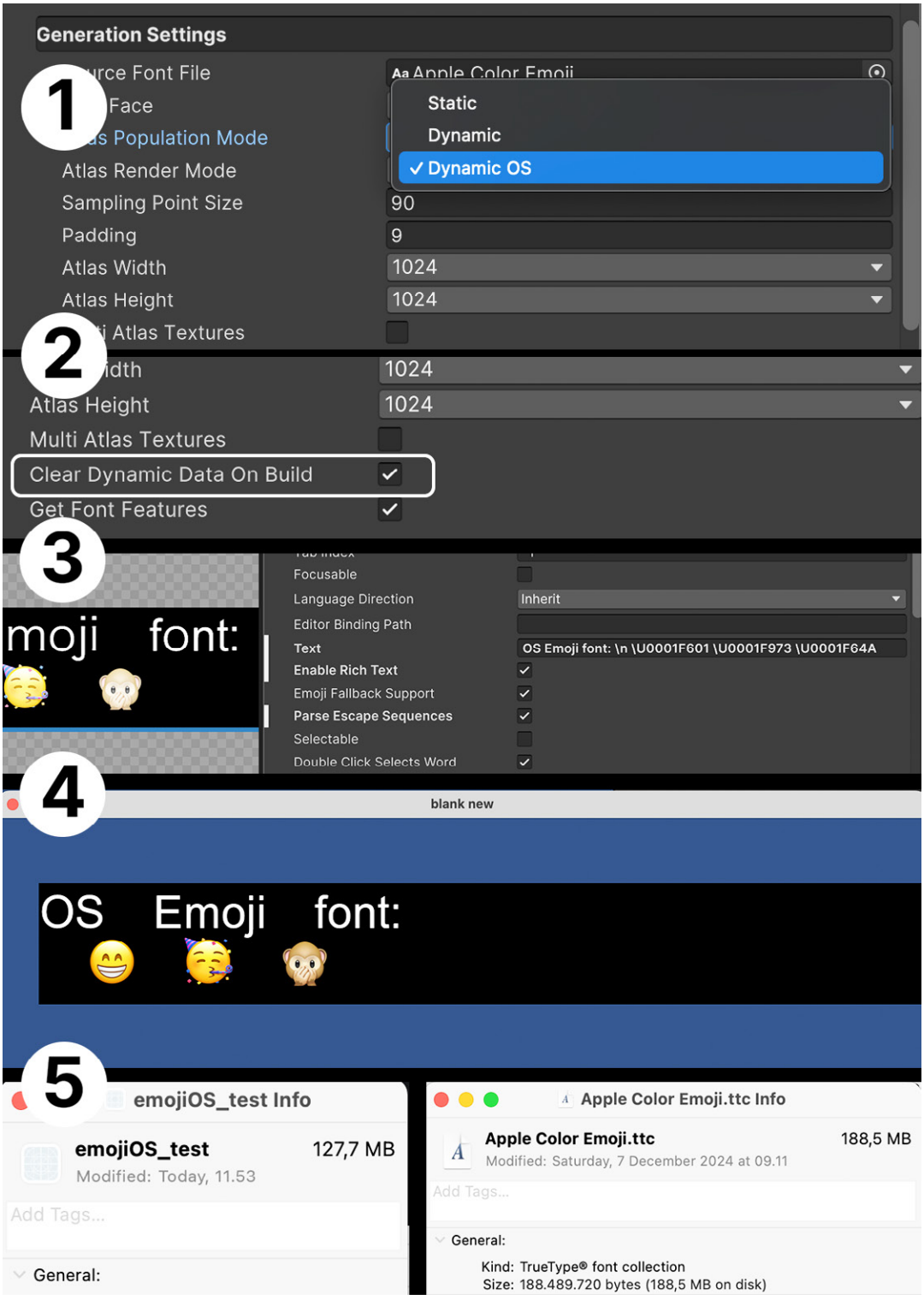
1. Создайте Font Asset на основе шрифта целевой системы. В iOS он называется Apple Emoji (этот шрифт используется в примере), а в Android - Noto Color Emoji; в настоящее время поддерживается только COLRv0. Убедитесь, что для Font Asset задан тип Color, затем установите для Atlas Population Mode значение Dynamic OS. В этом режиме не требуется включать исходный шрифт в ассет, что экономит место.

2. Убедитесь, что для Font Asset включён параметр Clean Dynamic Data On Build.

3. В UI Builder включите Parse Escape Sequences и введите нужные эмодзи с помощью клавиатуры эмодзи macOS или Windows либо в формате UTF. Например, смайлик можно задать как \U0001F601. UTF-код каждого эмодзи указан в Character Table объекта Font Asset.

4. Сборка, запущенная в macOS, отображает эмодзи с помощью шрифта операционной системы.

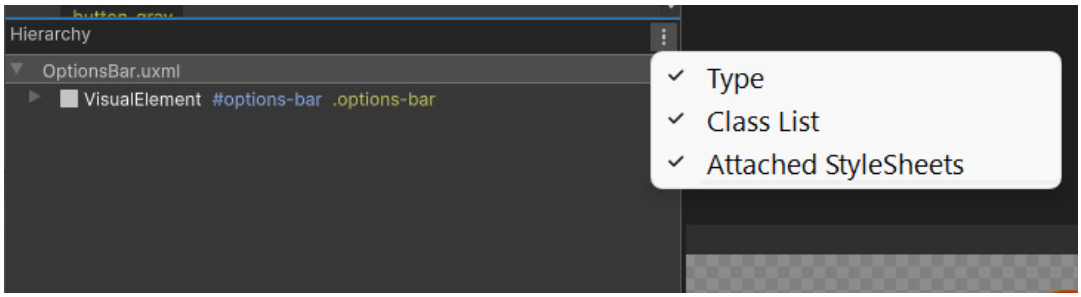
5. В нашем тесте размер сборки оказался меньше размера отдельного шрифта эмодзи. Это подтверждает, что шрифт не был включён в проект, хотя продолжал использоваться для отображения соответствующих эмодзи.



Отображайте дополнительную информацию о Visual Element в UI Builder

Нажмите кнопку с вертикальным многоточием (⋮) в заголовке Hierarchy, чтобы отобразить дополнительные сведения об элементах UI.

В панели Hierarchy рядом с типом элемента появятся дополнительные сведения. Если включить соответствующие параметры, будут показаны селектор имени #options-bar и селектор класса стиля .options-bar.



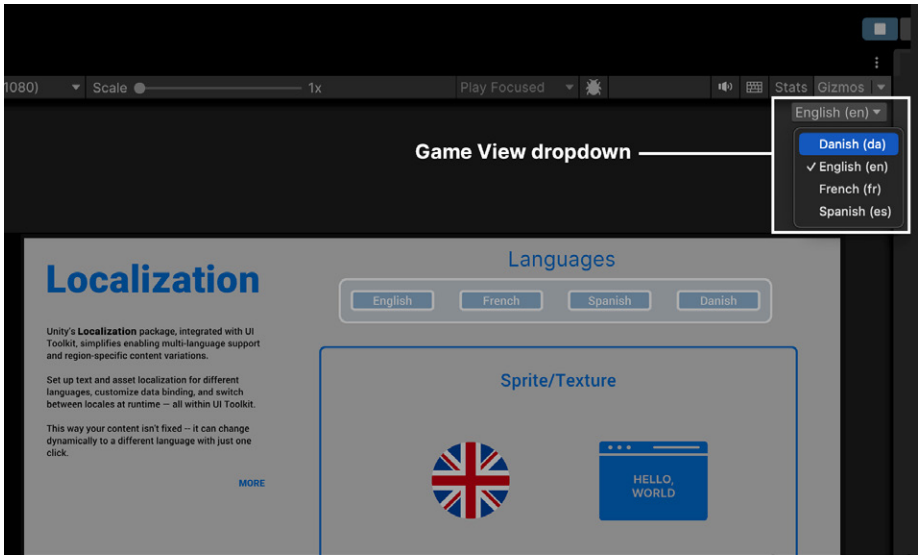
Фильтрация различных селекторов в панели Hierarchy

Можно заметить, что некоторые селекторы начинаются с префикса .unity-. Это стандартные стили, применяемые ко всем элементам. Любые явно заданные селекторы переопределяют эти значения.

Охватывайте больше рынков, внедряя локализацию на раннем этапе

Процесс локализации можно упростить, интегрировав пакет Localization с UI Toolkit. Такая интеграция позволяет предоставлять игрокам содержимое для их региона независимо от того, где они находятся.

Используйте раскрывающийся список Game View Locale для предварительного просмотра UI на разных языках и проверки правильности отображения элементов в каждой локали.



Предварительный просмотр локализации с помощью раскрывающегося списка Game View Locale

Стилизуйте весь интерфейс с помощью градиентов

В UI Toolkit градиенты применяются с помощью тега <gradient>. Убедитесь, что параметр Rich Text включён: результат будет виден в UI Builder и в Game view.

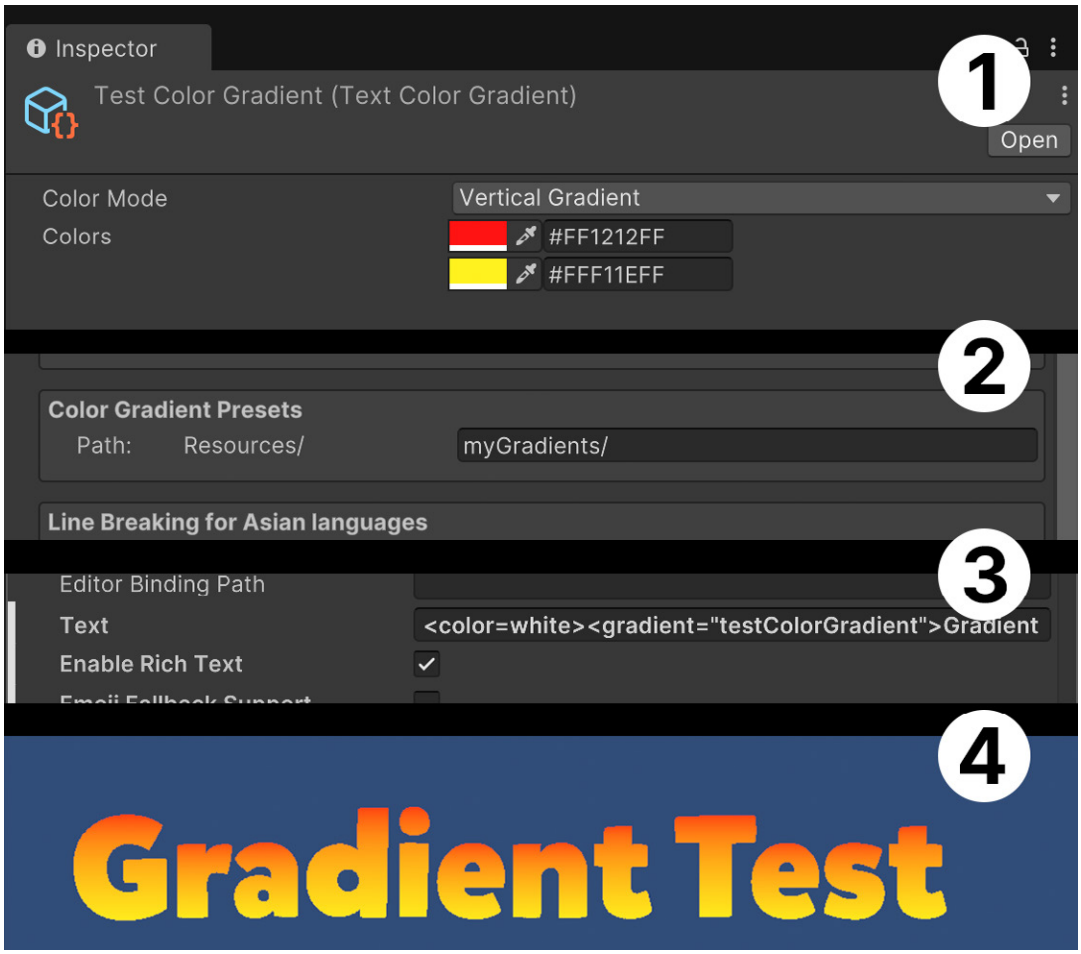
1. Создайте ассет градиента через Create > Text Core > Gradient Color. Поместите файл в Assets/Resources или в любую вложенную папку Resources.

2. Создайте ассет Text Settings, на который будет ссылаться Panel Settings. Найдите в нём параметр Color Gradient Presets и укажите папку или вложенную папку Resources, где находится ассет градиента.

3. В UI Builder добавьте следующие теги Rich Text:
<color=white><gradient="testColorGradient">Gradient Test</gradient></color>.

   Тег color возвращает белый цвет шрифта, чтобы градиент выглядел правильно. Имя в теге gradient должно совпадать с именем ассета, созданного на шаге 1. Убедитесь, что параметр Rich Text включён.

4. Результат изменений можно увидеть в UI Builder или в Game view.



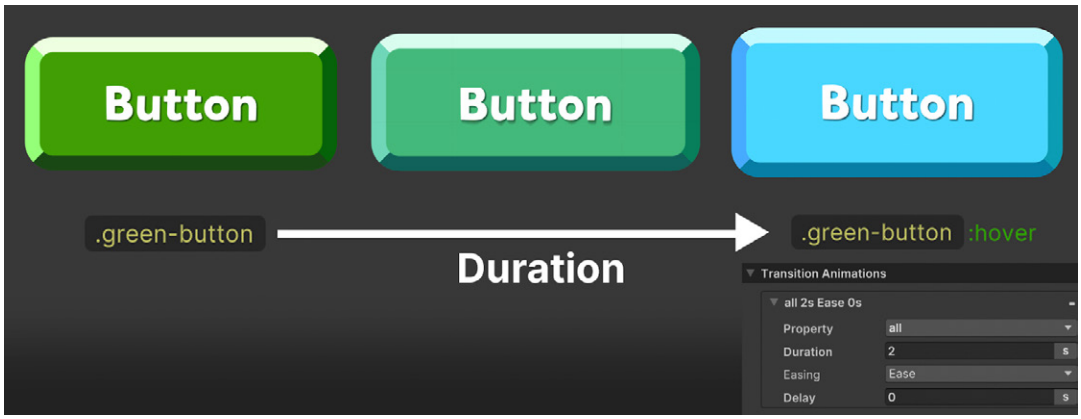
Использование тега <gradient> в UI Toolkit

Анимируйте UI с помощью переходов USS

Для визуальных элементов анимацию можно реализовать без дополнительного кода, поскольку у псевдоклассов (:active, :inactive, :hover и т. д.) могут быть собственные селекторы. Когда псевдокласс вызывает изменение стиля, заданные переходы автоматически анимируют это изменение.

Например, кнопка может увеличиваться или уменьшаться при наведении (:hover) или нажатии (:active), а элементы - постепенно исчезать или становиться невидимыми в ответ на действия пользователя и другие события.

Такие смены состояния обычно вызываются действиями пользователя, однако псевдоклассы :enabled и :disabled можно менять программно. Это вручную запускает связанные с ними анимации USS и позволяет вызывать анимацию из кода в нужный момент.

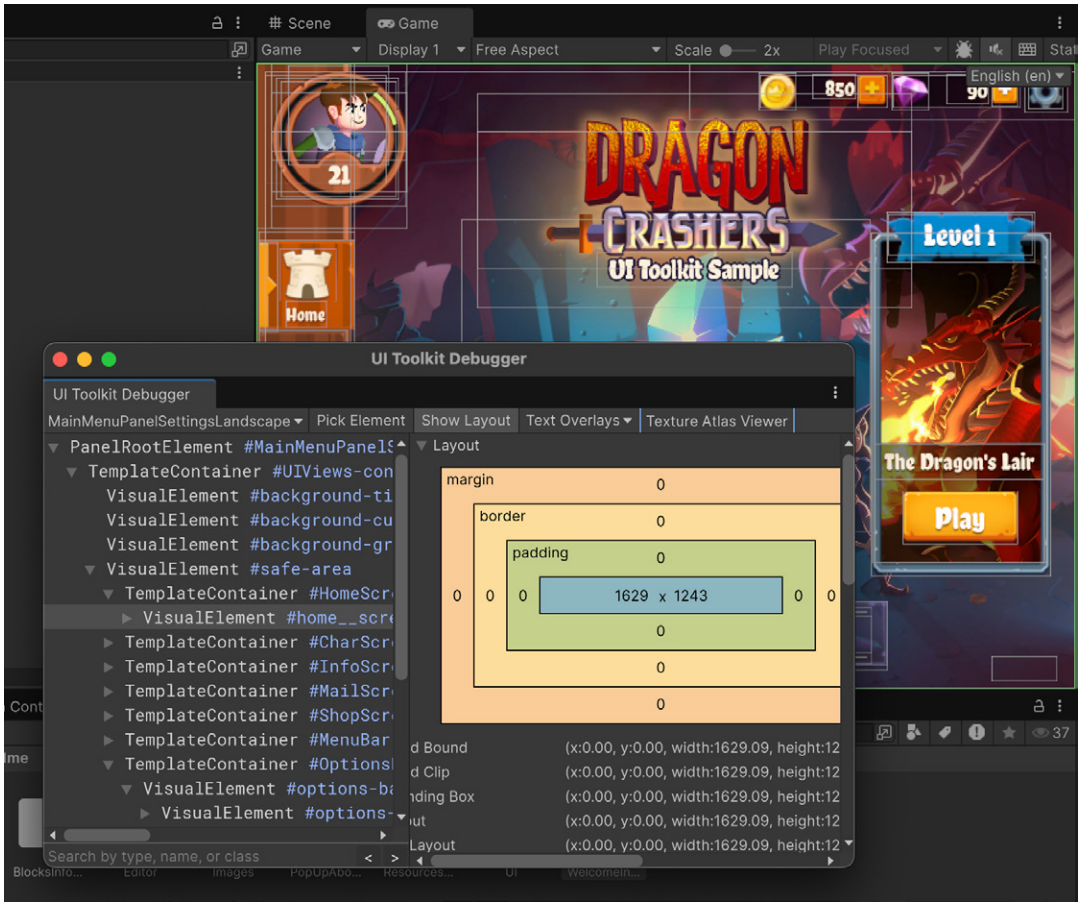


Пример анимации перехода USS

Показывайте рамки вычисленных стилей в редакторе

Ограничивающие рамки помогают выявлять ошибки компоновки и выравнивания, а также интерактивно отлаживать структуру UI в редакторе.

Чтобы отобразить вычисленные ограничивающие рамки средствами UI Toolkit, выберите Window > UI Toolkit > Debugger и откройте UI Toolkit Debugger. Затем инструментом Pick Element выберите нужный элемент UI и включите Show Layout. Поверх UI в Game view появятся ограничивающие рамки и сведения о компоновке.

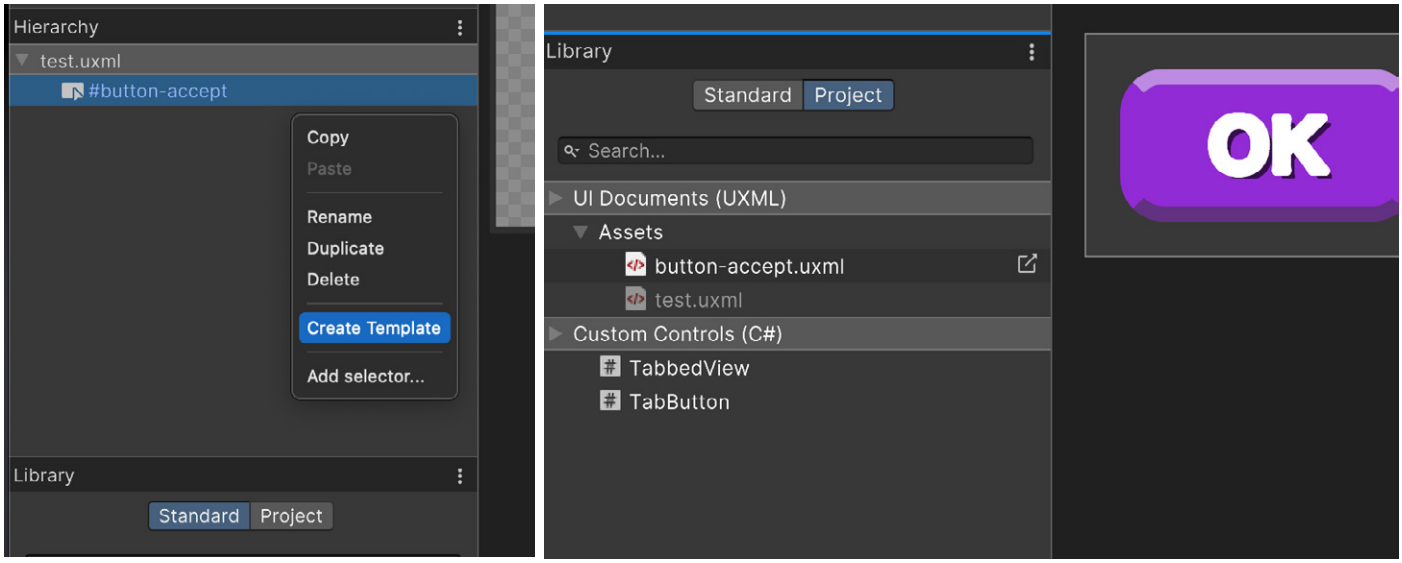


UI Toolkit Debugger содержит несколько удобных средств отладки UI.

Повторно используйте файлы UXML как шаблоны, чтобы ускорить работу

Файлы UXML можно использовать аналогично префабам. Например, в проекте может быть UXML-макет со значком предмета и счётчиком количества, который требуется многократно создавать в инвентаре.

Щёлкните любой UXML-файл правой кнопкой мыши и выберите создание Template. Затем шаблон можно добавить к любому другому визуальному элементу в панели Hierarchy или создать его экземпляр из кода. После создания шаблон доступен в Library и в окне Project.



Шаблоны - это повторно используемые UXML-файлы; они доступны в панели Library на вкладке Project

Дополнительные ресурсы

Загрузите электронную книгу «Создание масштабируемого и производительного UI с помощью UI Toolkit в Unity 6», чтобы получить подробные рекомендации по разработке интерфейсов для самых разных устройств.

К электронной книге прилагается демонстрационный проект. UI Toolkit Sample - Dragon Crashers доступен в Unity Asset Store. Этот проект показывает, как применять UI Toolkit в собственных приложениях. Демонстрация содержит полнофункциональный интерфейс для фрагмента двухмерной мини-RPG Dragon Crashers и использует рабочий процесс UI Toolkit в Unity 6 во время выполнения.

Кроме того, ознакомьтесь с проектом QuizU в Unity Asset Store и сопутствующими статьями:

- Демонстрационный проект UI Toolkit QuizU

- QuizU: паттерны состояний для управления ходом игры

- QuizU: управление экранами меню в UI Toolkit

- QuizU: паттерн Model View Presenter

- QuizU: обработка событий в UI Toolkit

- QuizU: советы по повышению производительности UI Toolkit

Рабочие процессы разработчика

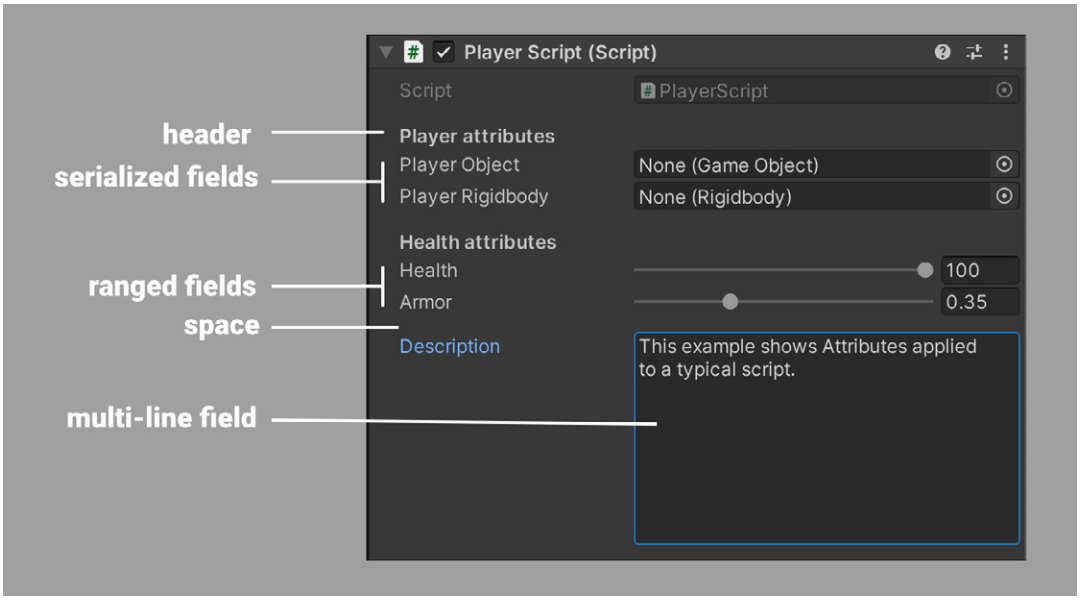
Класс Awaitable

В Unity 6 появился класс Awaitable - лёгкий тип без выделения памяти, специально разработанный для поддержки асинхронных рабочих процессов C# на основе async/await в Unity. Он служит производительной альтернативой корутинам и упрощает написание асинхронного кода, который корректно встраивается в покадровый цикл обновления Unity.

```
using UnityEngine;
using System.Threading.Tasks;

public class LogWithDelay : MonoBehaviour
{
    private async void Start()
    {
        Debug.Log(“Message 1”);
        await Task.Delay(1000); // Wait 1 second

        Debug.Log(“Message 2”);
        await Task.Delay(1000); // Wait another second
    }
}
```

Расширяйте возможности окна Inspector с помощью атрибутов

Unity предоставляет множество атрибутов, которые можно размещать перед классом, свойством или функцией, чтобы задать особое поведение: например, добавить в Inspector заголовок, отступ или поле с ограниченным диапазоном значений.



Атрибуты, влияющие на поля окна Inspector

В C# имена атрибутов заключаются в квадратные скобки. Ниже перечислены некоторые распространённые атрибуты, которые можно добавлять в скрипты.

Атрибут	Описание	Пример
SerializeField	Заставляет Unity сериализовать закрытое поле и делает его видимым в окне Inspector.	`[SerializeField]` `private GameObject m_myObject;`

Range	Этот атрибут задаёт допустимый диапазон для переменной float или int. В окне Inspector поле отображается в виде ползунка.	`[Range(1,6)]`
		`public int IntegerRange;`
		
		`[Range(0.2f, 0.8f)]`
		`public float m_floatRange;`

HideInInspector	Скрывает переменную в окне Inspector, не отключая её сериализацию.	`[HideInInspector]` `public Int p = 5;`

RequireComponent	Автоматически добавляет необходимые компоненты как зависимости, помогая избежать ошибок настройки.
	
	Примечание. Этот атрибут выполняет проверку только в момент добавления компонента к объекту GameObject.
		`// PlayerScript requires the GameObject to have a Rigidbody`
		`[RequireComponent(typeof(Rigidbody))]`
		`public class PlayerScript: Monobehaviour`
		`{`
		`  private Rigidbody m_rBody;`
		
		`  void Start()`
		`  {`
		`  m_rBody =`
		`  GetComponent<Rigidbody>();`
		`  }`
		`}`

Tooltip	Показывает всплывающую подсказку, когда пользователь наводит указатель мыши на поле в окне Inspector.	`public class PlayerScript: Monobehaviour`
		`{`
		`  [Tooltip(“Health value between 0 and 100.”)]`
		`  int m_health = 0;`
		`}`

Space	Добавляет небольшой отступ между полями без дополнительного текста, визуально разделяя их.	`[Space(10)] // 10 pixel of spacing added` `int p = 5;`

Header	Добавляет полужирный заголовок и отступ, помогая упорядочить переменные в окне Inspector. Применяйте атрибут только к первому полю соответствующей группы.	`public class PlayerScript: Monobehaviour`
		`{`
		`  [Header(“Health Settings”)]`
		`  private int m_health = 0;`
		`  private int m_maxHealth = 100;`
		
		`  [Header(“Shield Settings”)]`
		`  private int m_shield = 0;`
		`  private int m_maxShield = 0;`
		`}`

Multiline	Делает строку редактируемой в многострочном текстовом поле. Необязательный параметр int задаёт количество строк.
	
	Совет. Используйте этот атрибут, чтобы добавлять в скрипты примечания для себя или других пользователей.
		`[Multiline]`
		`public string textToEdit;`
		
		`[Multiline(20)]`
		`public string m_moreTextToEdit;`

SelectionBase	Полезен для выбора пустого объекта GameObject, дочерние объекты которого содержат меши. Добавьте этот атрибут к любому компоненту базового объекта. Тогда при выборе объектов в редакторе будет выбран GameObject с атрибутом [SelectionBase], а не его дочерние объекты.	`// add this to the base GameObject`
		`[SelectionBase]`
		`public class PlayerScript: Monobehaviour`
		`{`
		
		`}`

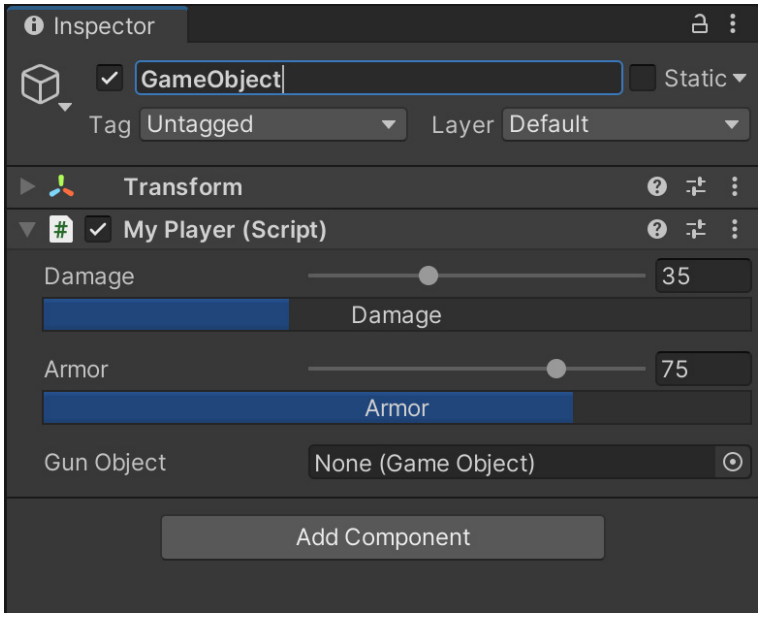
ColorUsage	Атрибут [ColorUsage] позволяет управлять тем, какие цвета можно выбирать в поле цвета. В зависимости от параметров можно включить HDR и отключить альфа-канал.	`public ColorUsageAttribute(bool showAlpha, bool hdr, float minBrightness, float maxBrightness, float minExposureValue, float maxExposureValue);`

RunOnce	Нужно автоматически выполнить функцию лишь один раз при запуске проекта? Простой способ - использовать статический метод с атрибутом [RuntimeInitializeOnLoadMethod]. Он подходит для однократной инициализации проекта, например загрузчика, как показано в примере QuizU.	`public RuntimeInitializeOnLoadMethodAttribute(RuntimeInitializeLoadType loadType);`

Это лишь малая часть множества доступных атрибутов. Нужно переименовать переменные, не потеряв их значения, или выполнить определённую логику без пустого объекта GameObject? Можно даже создать собственный PropertyAttribute и определить пользовательские атрибуты для переменных скрипта. Полный список атрибутов приведён в Scripting API.

Создавайте пользовательские окна и окна Inspector

Одна из самых мощных возможностей Unity - расширяемый редактор. Для создания интерфейсов редактора, включая пользовательские окна и окна Inspector, рекомендуется использовать пакет UI Toolkit.



Пользовательский редактор изменяет представление скрипта MyPlayer в окне Inspector.

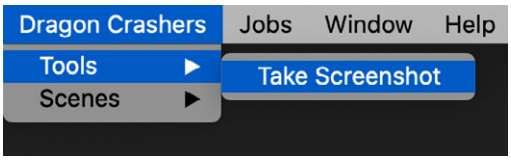
Подробнее о реализации пользовательских скриптов редактора с помощью UI Toolkit или IMGUI см. в разделе «Создание пользовательских интерфейсов (UI)». Краткое введение в UI Toolkit доступно в руководстве «Начало работы со скриптами редактора».

Создавайте пользовательские меню

Unity позволяет легко настраивать меню редактора и их пункты с помощью атрибута MenuItem. Его можно применить к любому статическому методу в скриптах.

Если некоторые функции проекта используются часто, оформите их как пункты меню. Так можно создать простой пользовательский интерфейс всего с одним модификатором PropertyAttribute.

```
using UnityEditor;
using UnityEngine;
using System;
using System.IO;

public class ScreenshotTaker
{
    [MenuItem("Dragon Crashers/Tools/Take Screenshot")]
    public static void TakeScreenshot()
    {
        if (!Directory.Exists("Screenshots"))
            Directory.CreateDirectory("Screenshots");

        ScreenCapture.CaptureScreenshot(string.Format("Screenshots/{0}.png",
            DateTime.Now.ToString("yyyy-MM-dd HH.mm.ss")));
    }
}
```



Атрибут MenuItem создаёт простой интерфейс для вызова статического метода Take Screenshot.

Ускорьте переход в режим Play

При переходе в режим Play проект запускается так же, как готовая сборка. Все изменения, внесённые в редакторе в режиме Play, сбрасываются после выхода из этого режима.

При каждом переходе в режим Play Unity выполняет два важных действия:

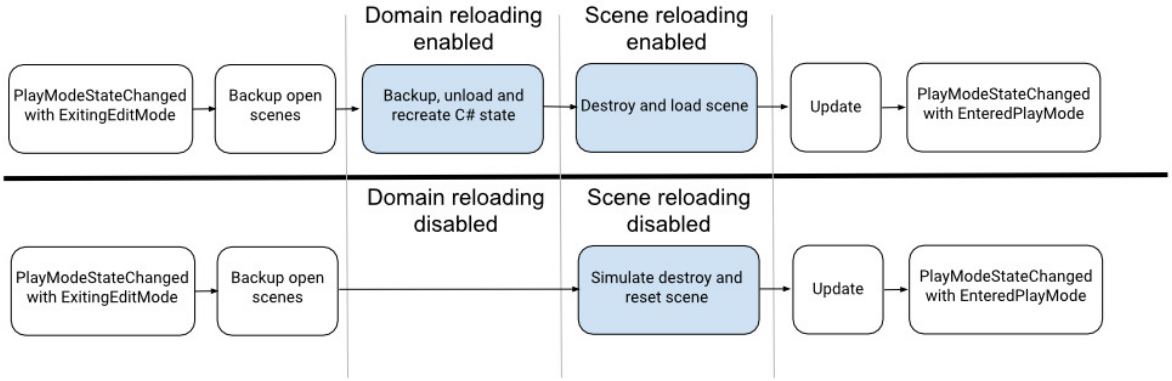
- Domain Reload: Unity сохраняет, выгружает и заново создаёт состояние скриптов.

- Scene Reload: Unity уничтожает сцену и загружает её заново.

По мере усложнения скриптов и сцен эти два действия занимают всё больше времени.

Если дальнейшие изменения скриптов не планируются, параметры Enter Play Mode Settings (Edit > Project Settings > Editor) помогут сократить время компиляции. Unity позволяет отключить Domain Reload, Scene Reload или оба действия, что ускоряет вход в режим Play и выход из него.

Помните: если вы планируете продолжать изменять скрипты, Domain Reload необходимо снова включить. Аналогично, после изменения иерархии сцены следует снова включить Scene Reload. Иначе возможны неожиданные результаты.



Результат отключения параметров Reload Domain и Reload Scene

Настраивайте стандартные шаблоны скриптов

Если при создании каждого нового скрипта вы вносите одни и те же изменения - например, сразу добавляете пространство имён или удаляете функцию события Update, - настройте исходный шаблон скрипта. Это сократит число нажатий клавиш и обеспечит единообразие в команде.

При создании нового скрипта или шейдера Unity использует шаблон из папки %EDITOR_PATH%\Data\Resources\ScriptTemplates:

— Windows: *C:\Program Files\Unity\Editor\Data\Resources\ScriptTemplates*

- Mac: /Applications/Hub/Editor/[version]/Unity/Unity.app/Contents/Resources/ScriptTemplates

Также доступны шаблоны шейдеров, других скриптов поведения и определений сборок.

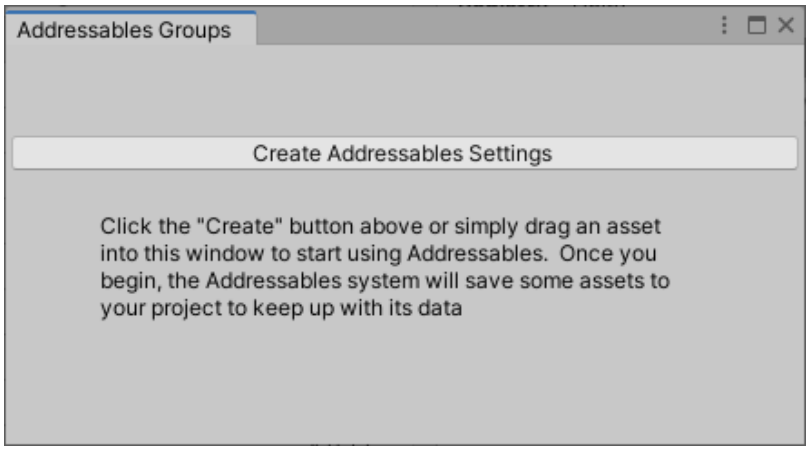
Чтобы использовать шаблоны скриптов только в конкретном проекте, создайте папку Assets/ScriptTemplates и скопируйте в неё шаблоны, которые должны переопределить стандартные.

Стандартные шаблоны скриптов можно изменить непосредственно для всех проектов, но перед этим обязательно сохраните резервные копии исходных файлов.

Доставляйте игрокам контент по запросу с помощью Addressables

Addressables и Asset Bundles - мощные средства, позволяющие разделить игру на логические блоки. Эти блоки можно экспортировать отдельно и при необходимости добавлять к основному исполняемому файлу.

Они позволяют загружать и выгружать ассеты, а также настраивать, собирать и загружать пакеты ассетов, которые затем можно доставлять игрокам по запросу. Система Addressables построена поверх Asset Bundles и сама разрешает зависимости и загружает пакеты.



До инициализации системы Addressables в проекте Unity

Если вы только начинаете работать с Addressables, ознакомьтесь со страницей Get started в документации Unity.

Советы по эффективному управлению ассетами:

- Используйте Addressables с самого начала разработки и регистрируйте каждый новый ассет как Addressable.

- Группируйте ассеты по тому, насколько часто они загружаются и используются вместе, а не по типу. Это улучшает использование памяти во время выполнения, сокращает время запуска и, как следствие, способствует удержанию игроков.

- Старайтесь создавать небольшие пакеты: это сокращает цепочки зависимостей и снижает расход памяти во время выполнения.

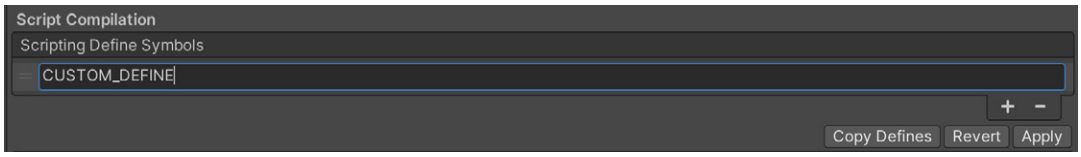
Подробнее см. в статье «Эффективное управление ассетами в Unity с помощью Addressables».

Создавайте условно компилируемый код с помощью директив препроцессора

Платформозависимая компиляция позволяет условно компилировать и выполнять код в зависимости от целевой платформы, версии Unity или скриптового бэкенда.

Это полезно при разработке кроссплатформенного кода, оптимизации поведения для конкретных устройств и работе с API, доступными только в определённых версиях.

При тестировании в редакторе можно задавать собственные директивы #define. Откройте панель Other Settings в Player settings и перейдите к параметру Scripting Define Symbols.



Параметр Scripting Define Symbols в разделе Script Compilation

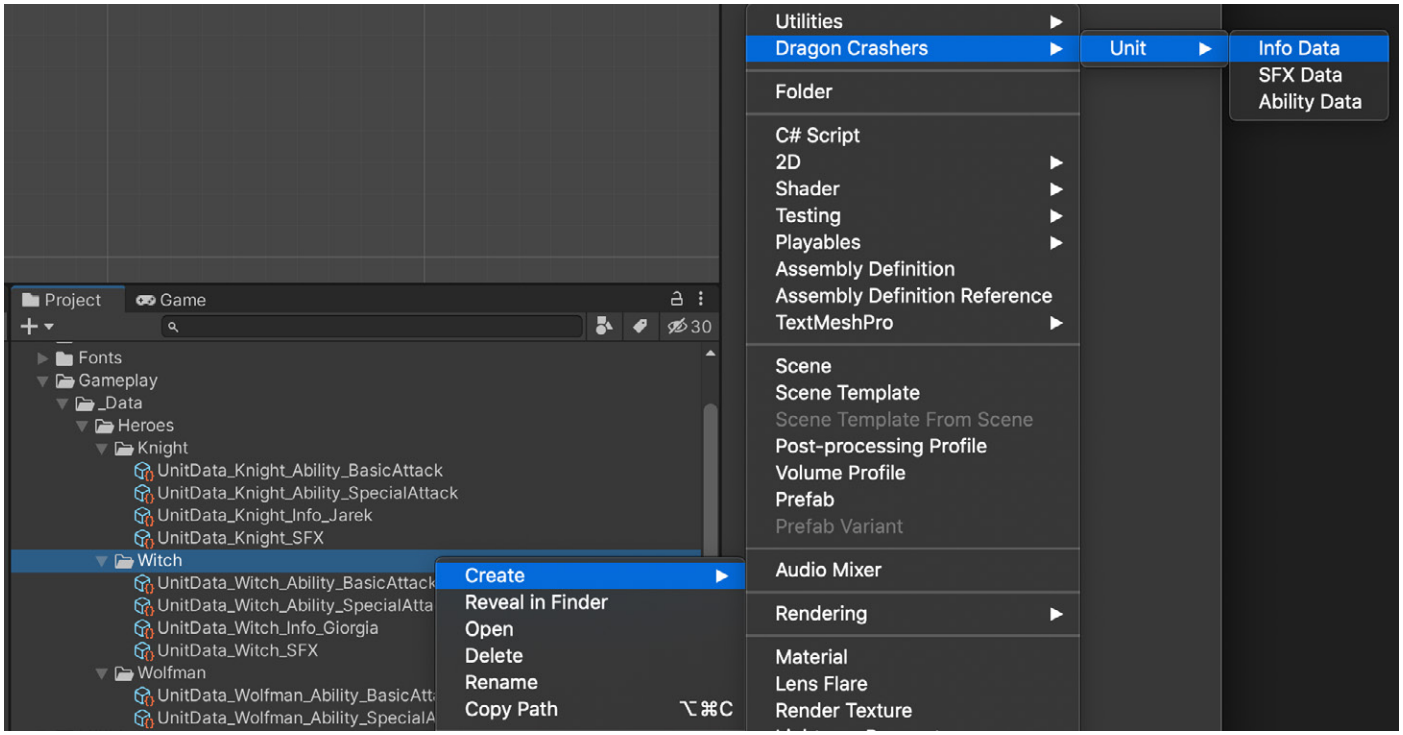
Отделяйте данные от логики с помощью ScriptableObject

ScriptableObject помогает поддерживать чистую архитектуру кода, отделяя данные от логики. Благодаря этому проще вносить изменения без нежелательных побочных эффектов, а код становится более тестируемым и модульным. Такие объекты также удобны при совместной работе с художниками и дизайнерами: они могут редактировать игровые данные, не изменяя код.

Dragon Crashers демонстрирует типичный сценарий использования. Класс UnitInfoData наследуется от ScriptableObject. Каждый его экземпляр содержит имя игрового персонажа, спрайт и параметры здоровья. Эти данные остаются неизменными в ходе игры, поэтому особенно хорошо подходят для хранения в ScriptableObject.

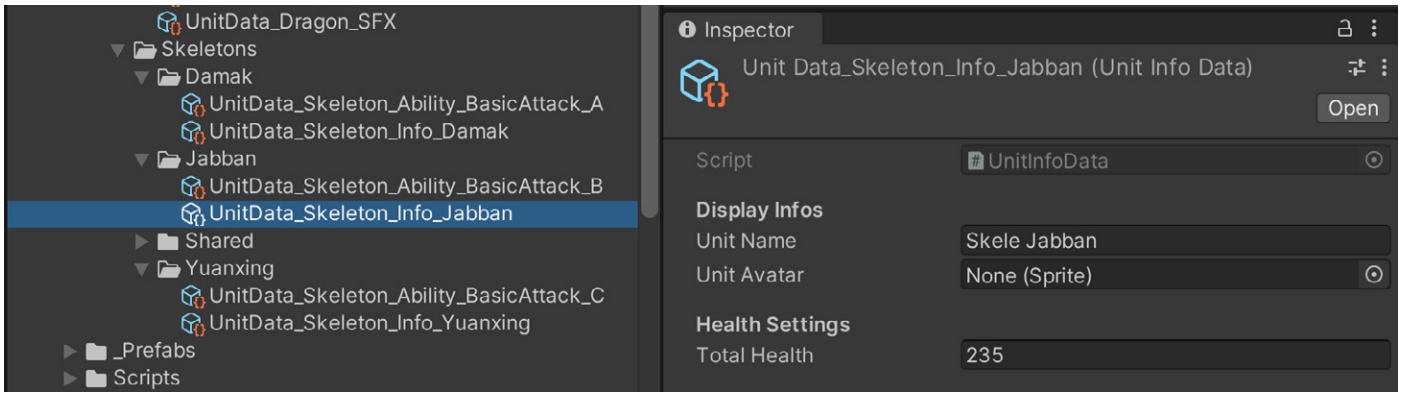
```
using UnityEngine;

namespace DragonCrashers
{
    [CreateAssetMenu(fileName = "Data_Unit_", menuName = "Dragon Crashers/Unit/Info Data", order = 1)]
    public class UnitInfoData : ScriptableObject
    {
        [Header("Display Infos")]
        public string unitName;
        public Sprite unitAvatar;

        [Header("Health Settings")]
        public int totalHealth;

    }
}
```

ScriptableObject определяет объект-контейнер данных.



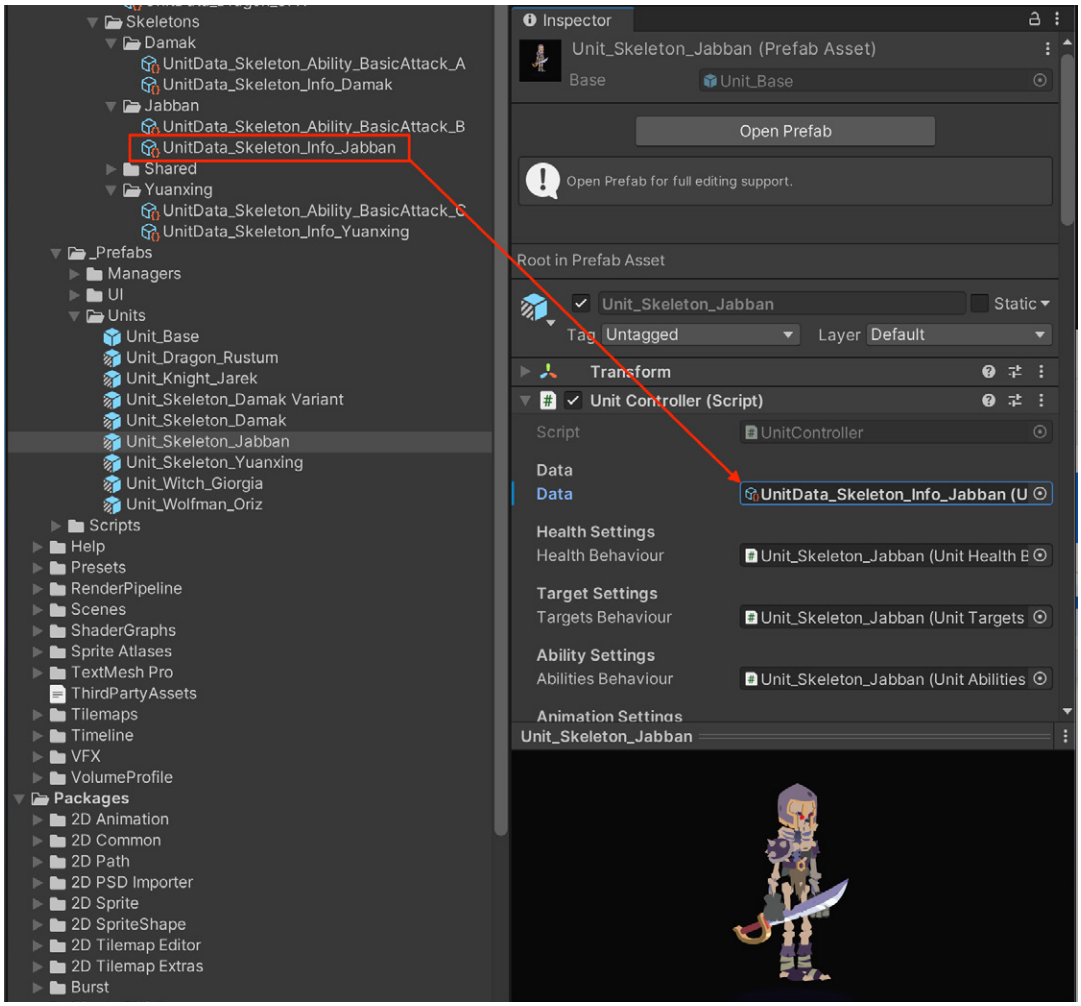
Атрибут CreateAssetMenu создаёт пункт контекстного меню для генерации ассета ScriptableObject. У каждого игрового персонажа есть дополнительные объекты ScriptableObject для звуковых эффектов и особых способностей.

После создания ассетов в окне Project задайте нужные значения в окне Inspector: Unit Name, Unity Avatar (Sprite) и Total Health.

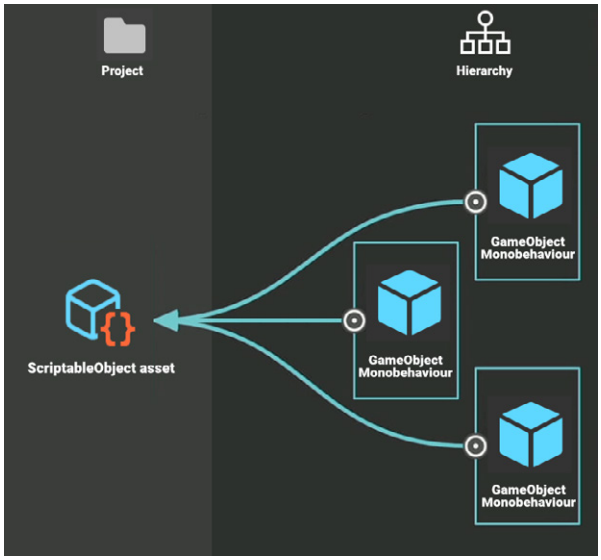


Задайте значения ассета ScriptableObject в окне Inspector. Во время игры они не изменяются.

Объект GameObject, в данном случае UnitController, может ссылаться на ассет ScriptableObject. Даже если сцена заполнится множеством игровых персонажей, данные ассета ScriptableObject не будут дублироваться, что экономит память.



Объект MonoBehaviour (показанный выше UnitController) ссылается на ассет ScriptableObject с данными проекта.



Экономьте память и поддерживайте порядок с помощью ScriptableObject. Статические данные и настройки достаточно задать в ассете проекта один раз, даже если у вас множество объектов GameObject.

Даже если добавить в сцену тысячу экземпляров префаба, все они будут ссылаться на одни и те же данные, хранящиеся в ассете. Достаточно один раз задать набор значений, чтобы гарантировать их согласованность.

По мере роста игры и появления новых типов юнитов просто создавайте дополнительные ассеты ScriptableObject и подставляйте нужные. Игровые данные можно поддерживать в актуальном состоянии, изменяя централизованно хранящиеся ассеты.

ScriptableObject не заменяют хранение постоянных данных в файлах сохранения приложения, где данные могут изменяться во время игры. Этот подход лучше подходит для статических игровых настроек и значений по умолчанию. В отличие от разбора JSON или XML, чтение ассета ScriptableObject не создаёт мусора (и к тому же выполняется быстрее).

Дополнительные материалы о ScriptableObject:

- Создание модульной архитектуры игры в Unity с помощью ScriptableObject

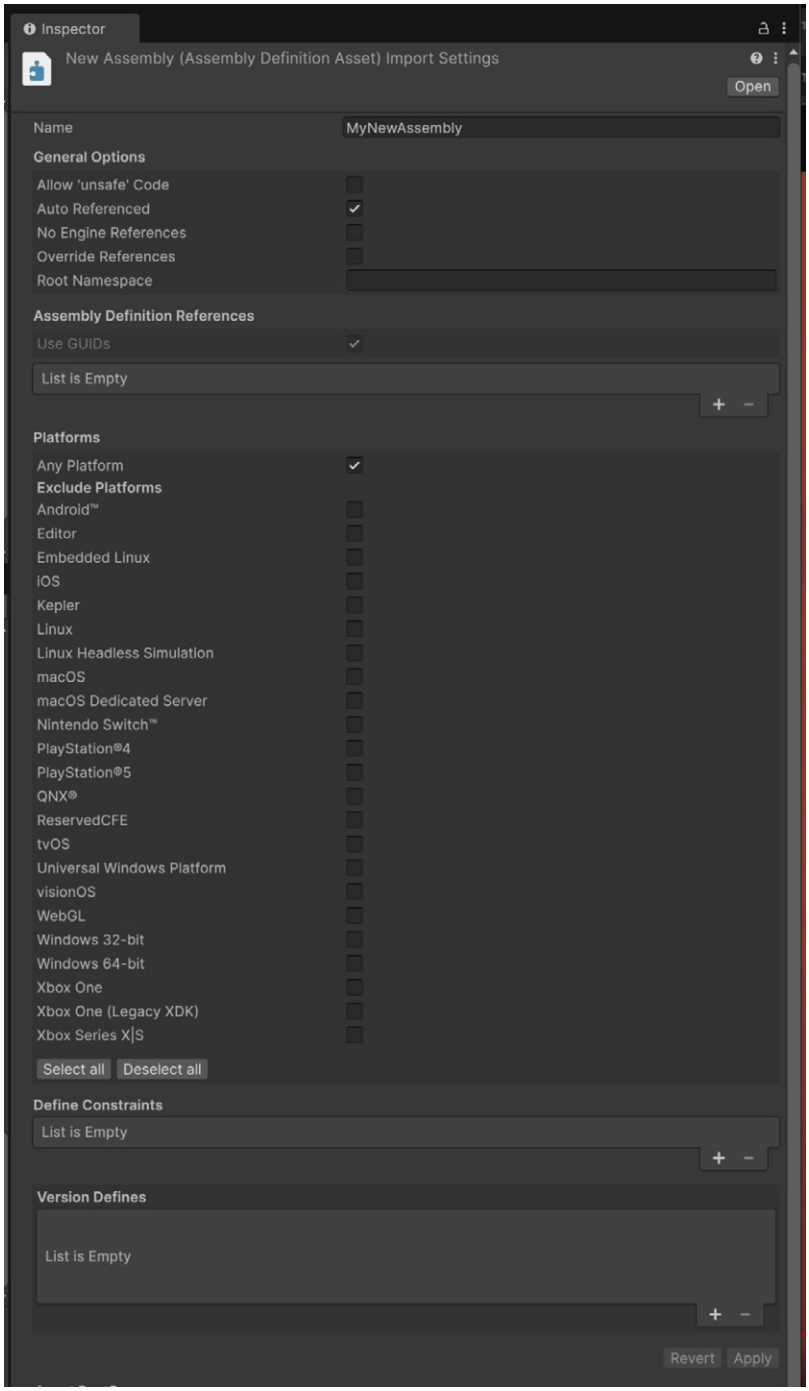
- Демонстрационный проект ScriptableObjects Paddle Ball

- Документация по ScriptableObject

Повышайте модульность скриптов с Assembly Definitions

Сборка - это скомпилированная библиотека кода C#, объединяющая связанные типы и ресурсы в единую логическую единицу. В Unity сборками можно управлять с помощью файлов Assembly Definition (.asmdef). Разделение скриптов на пользовательские сборки повышает модульность и возможность повторного использования, а также сокращает время компиляции. При этом скрипты не добавляются автоматически в стандартные сборки, а доступ к другим скриптам можно ограничить.

Если вы упорядочиваете проект с помощью Assembly Definitions, но скрипты Editor попадают в билды, создайте Assembly Definition в папке Editor и укажите для неё только платформу Editor.



Настройки Assembly Definitions в Inspector

Переходите на Input System

Если вы ещё не перешли, обратите внимание на пакет Input System - более новую и гибкую систему по сравнению с Input Manager, которая позволяет управлять содержимым Unity с помощью устройств ввода любого типа. Её называют пакетом Input System или просто Input System. Она также поддерживает переназначаемое управление, ассеты Input Action и более чёткое разделение логики ввода и игрового процесса, что даёт заметные преимущества перед устаревшей системой.

Для начала ознакомьтесь со следующими материалами:

- Ускоренное прототипирование мобильных игр с Input System в Unity 6 | Unite 2024

- Начало работы с Input System

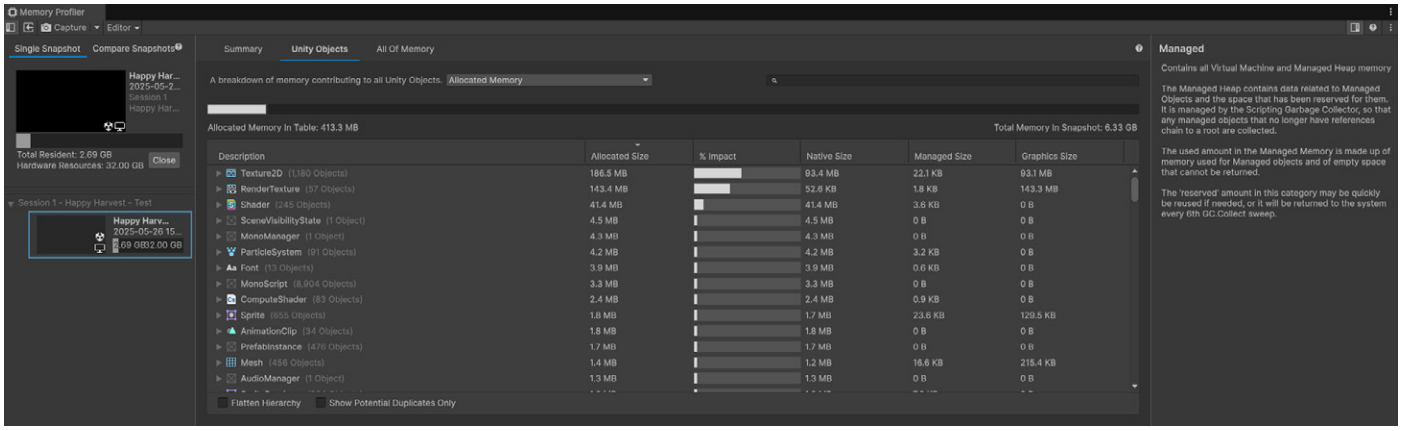
- Серия из семи видеоуроков по Unity Input System

Инструменты профилирования

Оптимизируйте использование памяти с помощью Memory Profiler

Memory Profiler позволяет фиксировать и анализировать использование памяти в проекте, чтобы выявлять утечки, уменьшать пиковое потребление и оптимизировать производительность во время выполнения. Делайте снимки памяти в ключевые моменты - например, при загрузке сцен или после длительных игровых сеансов - и сравнивайте их, чтобы находить объекты, которые не освобождаются должным образом.

Создавая ресурсы в коде, возьмите за правило присваивать им имена для Memory Profiler. Кроме того, не забывайте освобождать всё, что выделили, чтобы избежать утечек.



Снимок памяти в Memory Profiler

Находите критичный код с помощью ProfilerMarker

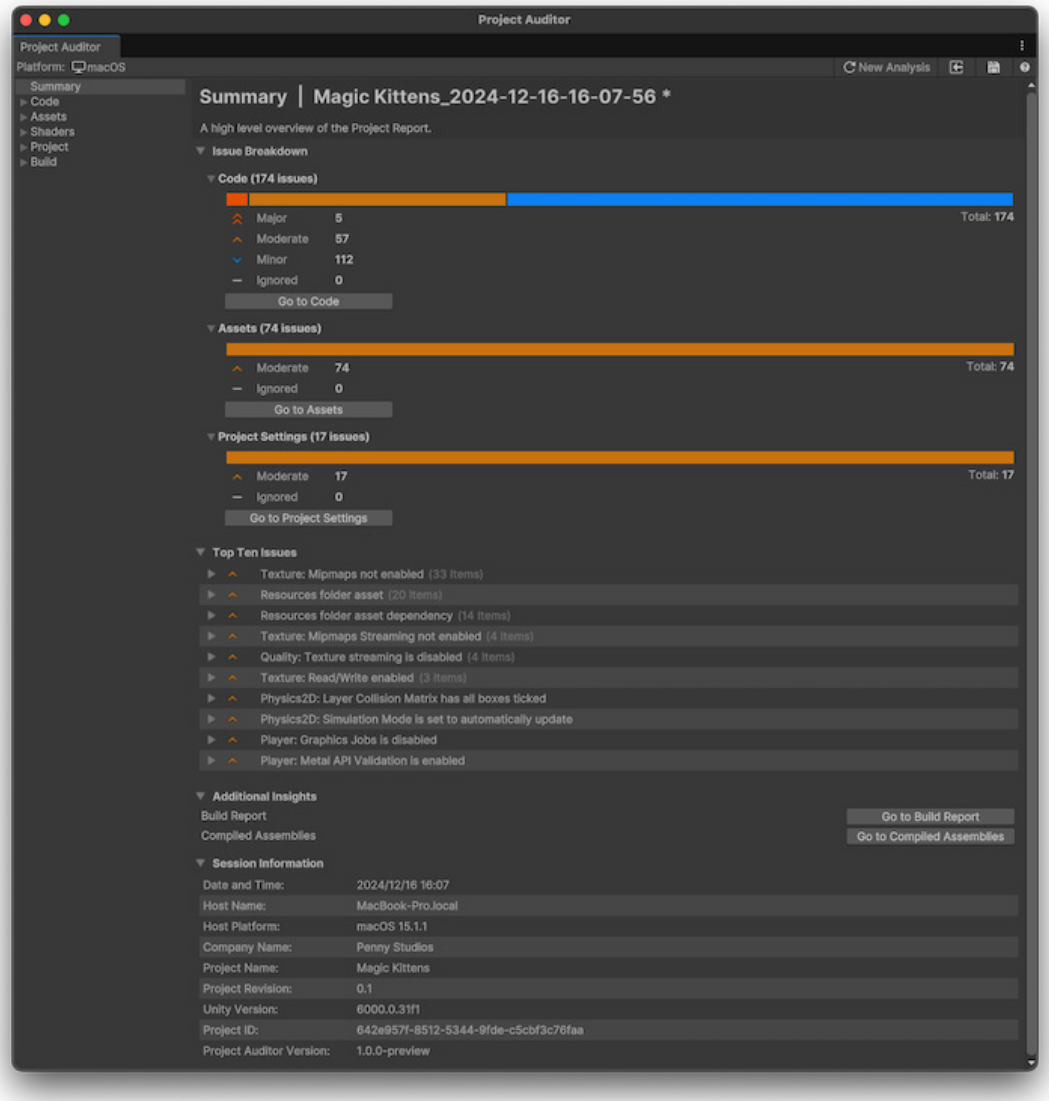
Вместо просмотра лишь общих данных о производительности под стандартными маркерами вроде BehaviourUpdate можно изолировать конкретные функции и точно измерить время их выполнения. С помощью ProfilerMarker отмечайте блоки кода скриптов, чтобы повысить детализацию профилирования. Результаты отображаются в CPU Profiler и могут записываться с помощью Unity Recorder. Так вы получите подробную картину затрат времени в конкретных участках кода, что упрощает поиск узких мест и оптимизацию.

```
using UnityEngine;
using Unity.Profiling;
public class UnityTips : MonoBehaviour {
   private static readonly ProfileMarker SetupProfileMarker = new Pro-
fileMarker("Setup");
   private static readonly ProfileMarker ExpensiveProfileMarker = new
ProfileMarker("Expensive");

   public void UpdateLogic() {
       SetupProfileMarker.Begin();
       // Setup your performance heavy things, Initializers, and so
on...
       SetupProfileMarker.End();
       using (ExpensiveProfileMarker.Auto()) { //This starts and ends
automatically
       // More expensive things here.
```

Проведите аудит производительности проекта

Используйте Project Auditor (доступный как пакет начиная с Unity 6.1), чтобы анализировать производительность проекта, соблюдать рекомендации и выявлять возможные проблемы и узкие места.

Всего за несколько щелчков можно просканировать весь проект и получить подробный отчёт о проблемах: затратных вызовах скриптов, неиспользуемых ассетах, чрезмерном количестве сущностей и многом другом.



Представление Summary в Project Auditor

Сформированные отчёты группируются по степени серьёзности: ошибки, предупреждения и информационные замечания. Благодаря этому легко сначала сосредоточиться на ошибках и предупреждениях, например на чрезмерном выделении памяти или слишком частой сборке мусора.

Обычно рекомендуется запускать Project Auditor на ключевых этапах разработки - например, перед контрольными точками, бета-выпусками и финальными билдами. Это позволяет заранее выявить узкие места производительности, неиспользуемые ассеты и устаревший код, не давая проблемам накапливаться по мере роста проекта.

Project Auditor можно настроить с помощью пользовательских правил и фильтров. Например, исключить из анализа заведомо неиспользуемые или экспериментальные скрипты и ассеты либо создать отдельные правила для целевых платформ, разрешения, сжатия текстур и других параметров проекта, чтобы они соответствовали установленным «бюджетам».

Кривые анимации

Управляйте интерполяцией с помощью пользовательской функции lerp

По умолчанию Mathf.Lerp(a, b, t) ограничивает коэффициент интерполяции t диапазоном от 0 до 1, поэтому результат не выходит за пределы значений a и b. Если требуется выход за верхнюю (t > 1) или нижнюю (t < 0) границу, используйте Mathf.LerpUnclamped(a, b, t). Эта функция даёт полный контроль над интерполяцией и позволяет создавать такие эффекты, как экстраполяция или движение по инерции.

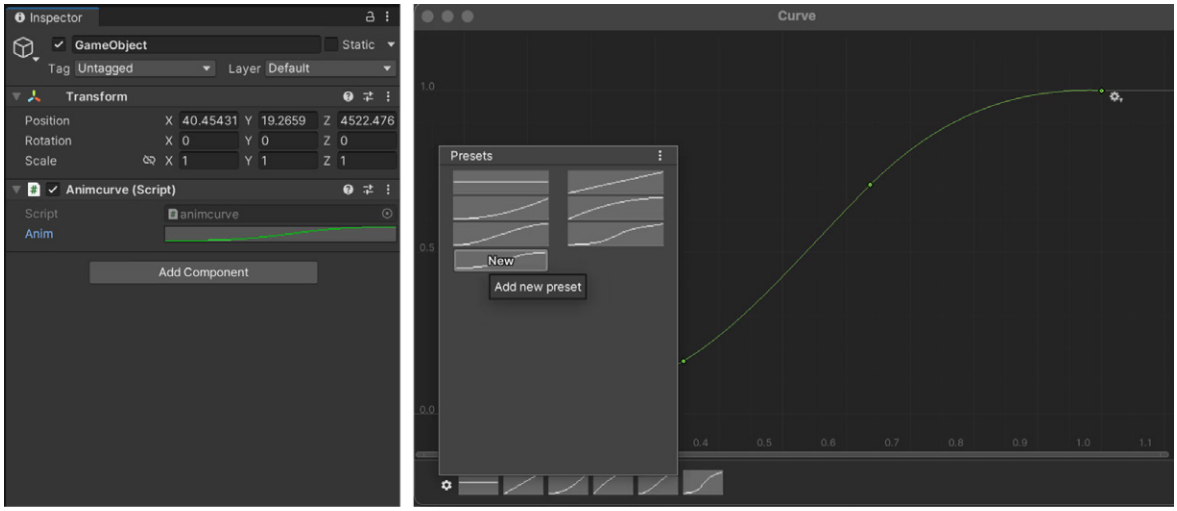
```
using UnityEngine;

public class LerpComparison : MonoBehaviour
{
    [SerializeField] private float start = 0f;
    [SerializeField] private float end = 10f;
    [SerializeField] private float t = 1.5f;

    private void Start()
    {
        // Clamps t to [0, 1]
        float clamped = Mathf.Lerp(start, end, t);
        // Uses full t value
        float unclamped = Mathf.LerpUnclamped(start, end, t);

        // Outputs 10
        Debug.Log($"Mathf.Lerp: {clamped} (t = {t})");
        // Outputs 15
        Debug.Log($"Mathf.LerpUnclamped: {unclamped} (t = {t})");
     }
}
```

Пример использования пользовательской линейной интерполяции в Unity

Используйте AnimationCurve не только для анимации

Объекты AnimationCurve обычно применяются для анимации значений свойств компонентов в AnimationClip, однако с их помощью можно динамически управлять любым значением типа float.

Кривые анимации можно редактировать в Inspector как открытые или сериализованные переменные. Их можно сохранять, экспортировать и загружать как в режиме редактирования, так и во время выполнения. Редактируемые касательные позволяют управлять формой кривой между ключами.



Свойство Animation Curve в Inspector: щелчок по нему открывает Curve Editor, где можно изменить кривую и сохранить её в собственной библиотеке, выбрав значок шестерёнки.

Дополнительные практические советы и примеры использования AnimationCurve в проекте приведены в публикации блога «Animation Curves - универсальный рычаг дизайна».

Снижайте нагрузку с помощью пулов объектов

Пул объектов - это паттерн проектирования, который помогает повысить производительность, уменьшая нагрузку на CPU от многократных вызовов создания и уничтожения объектов. Вместо этого уже существующие GameObject можно использовать повторно.

Способ применения пулов объектов зависит от конкретного приложения. Общее полезное правило: профилируйте код всякий раз, когда создаёте большое количество экземпляров объектов, поскольку это может вызвать пик сборки мусора.

Если вы обнаружили значительные пики, из-за которых игровой процесс может начать подтормаживать, рассмотрите применение пула объектов. Учтите, что управление несколькими жизненными циклами пулов усложняет кодовую базу. Кроме того, если создать слишком много пулов заранее, можно зарезервировать память, которая в действительности игре не нужна.

Подробнее о пулах объектов рассказывается в электронной книге «Совершенствуйте код с помощью паттернов проектирования и SOLID» и в сопутствующем демонстрационном проекте, бесплатно доступном в Unity Asset Store.

Дополнительные материалы

- Руководство по стилю C# для чистого и масштабируемого игрового кода (издание для Unity 6)

- Справочник гейм-дизайнера Unity

- Создание модульной архитектуры игры в Unity с помощью ScriptableObject

- Эффективное управление ассетами в Unity с помощью Addressables

- Что нужно знать о Build Profiles в Unity 6

IDE и отладка

Приостанавливайте выполнение с помощью Debug.Break

Если приложение сложно приостановить вручную, а вам нужно проверить определённые значения в Inspector, вызовите Debug.Break, чтобы остановить выполнение в нужном месте кода.

Заменяйте проверку if вызовом Debug.Assert

Debug.Assert проверяет условие во время выполнения и выводит сообщение об ошибке в Console, если условие возвращает false. В отличие от Debug.Log, который выполняется всегда, утверждения предназначены для выявления неожиданных состояний и помогают эффективнее проверять предположения, заложенные в коде.

```
// You can save the if statement in release...
if (health > maxhealth)
{
   Debug.LogError("Current health is greater than maxhealth!", this);
}
//... by using an assertion
Debug.Assert(health < maxhealth, "Current health is greater than max-
health!", this);
```

Используйте Debug.Log с контекстом

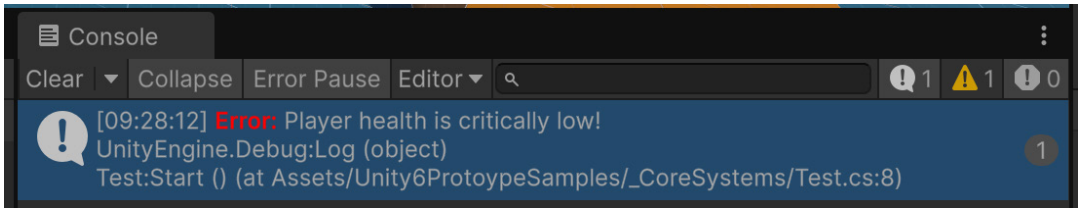
При вызове Debug.Log вторым параметром можно передать объект - обычно GameObject или компонент. Тогда сообщение журнала связывается с этим объектом в Console: если щёлкнуть по сообщению, Unity выделит соответствующий объект в Hierarchy.

```
Debug.Log(“Enemy spawned”, gameObject);
```

Выделяйте важные сообщения с помощью Rich Text

Console в Unity поддерживает в сообщениях Debug.Log часть тегов Rich Text, например <b>, <i> и <color>. С их помощью можно выделять, окрашивать или подчёркивать отдельные фрагменты вывода журнала, чтобы важные сообщения были заметнее во время разработки.

```
Debug.Log(“<b><color=red>Error:</color></b> Player health is critically
low!”);
```



Удаляйте вызовы Debug.Log из релизных билдов

Unity не удаляет API журналирования Debug из билдов, не являющихся Development Build, автоматически. Оберните вызовы Debug.Log в собственные методы и пометьте их атрибутом [Conditional].

Если удалить соответствующий Scripting Define Symbol из Player Settings, все вызовы Debug.Log будут исключены при компиляции. Результат такой же, как при оборачивании вызовов в блоки препроцессора #if... #endif.

Пример приведён в руководстве по общей оптимизации.

Диагностируйте физику, визуализируя raycast-запросы

Возникли проблемы с физикой? Debug.DrawLine и Debug.DrawRay помогут визуализировать raycast-запросы, рисуя линию между заданными начальной и конечной точками.

```
    void Start()
     {
         // draw a 5-unit white line from the origin for 2.5 seconds
         Debug.DrawLine(Vector3.zero, new Vector3(5, 0, 0), Color.
white, 2.5f);
     }
```

Используйте Debug.isDebugBuild для Development Build

С помощью Debug.isDebugBuild можно проверить, запущено ли приложение как Development Build. Это позволяет условно выполнять код, предназначенный только для отладки, - журналирование, диагностику или тестовые утилиты - не затрагивая релизные билды.

```
if (Debug.isDebugBuild)
{
    Debug.Log(“Running in Development Build mode.”);
}
```

Настройте Application.SetStackTraceLogType

Используйте Application.SetStackTraceLogType или соответствующие флажки в PlayerSettings, чтобы определить, для каких типов сообщений журнала следует добавлять трассировку стека. Трассировка стека полезна, но работает медленно и создаёт мусор.

Настройки Stack Trace в PlayerSettings окна Editor

Настройте журналирование под свои задачи

API Logger позволяет создавать и настраивать собственные экземпляры Logger для более сложного или модульного журналирования. Можно пользоваться встроенным Debug.unityLogger, однако собственный логгер даёт более точный контроль над форматированием, фильтрацией и каналами вывода:

```
var logger = new Logger(Debug.unityLogger.logHandler);
logger.Log(LogType.Log, “Custom log message”);
```

Ускорьте работу с помощью сочетаний клавиш Visual Studio Code

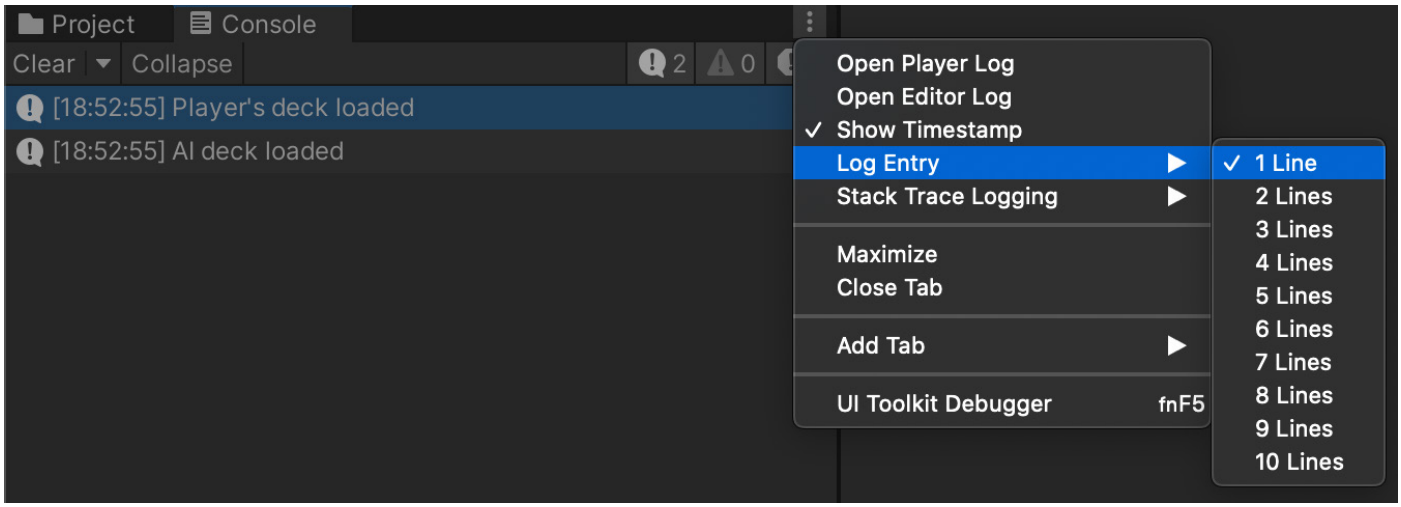
Если вы выбрали Visual Studio Code в качестве IDE, вам могут пригодиться следующие сочетания клавиш:

— Windows

— Mac

Настройте Console Log Entry для удобства чтения

По умолчанию Console Log Entry отображает две строки. Чтобы сделать вывод удобнее, можно выбрать одну или несколько строк в соответствии со своими предпочтениями (см. изображение ниже).



Параметры Console Log Entry позволяют задать количество строк в сообщении журнала.

Дополнительные материалы

- Отладка игрового кода с помощью Roslyn Analyzers
- Запуск автоматизированных тестов для игр с помощью Unity Test Framework
- Ускорение процесса отладки с помощью Microsoft Visual Studio Code
- Отладка кода с помощью Microsoft Visual Studio 2022
- Советы по тестированию и обеспечению качества проектов Unity

Рабочие процессы DevOps

Советы по Unity Version Control

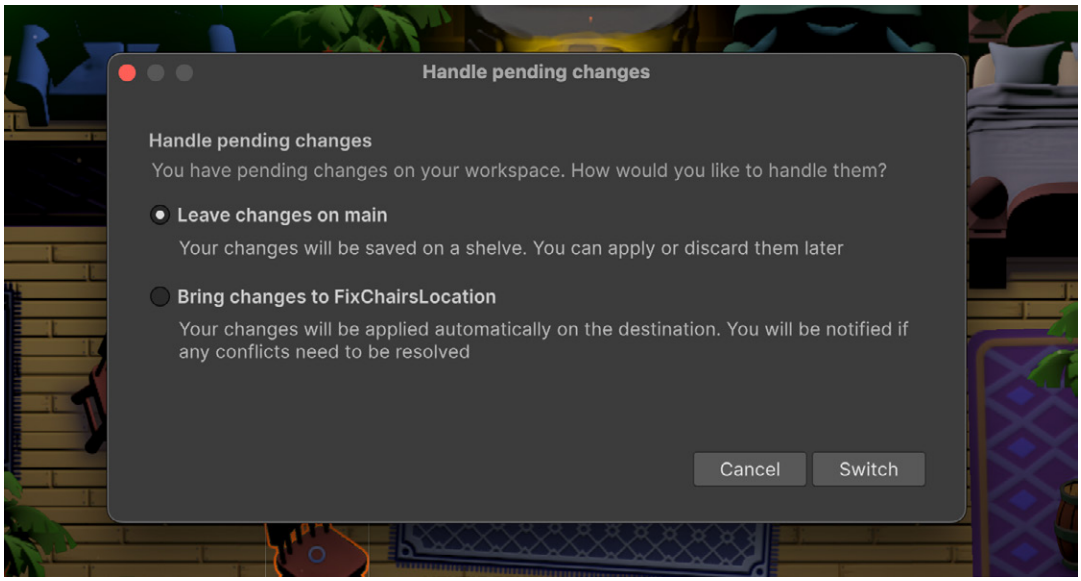
Unity Version Control (UVCS) - масштабируемая, независимая от игрового движка система контроля версий и управления исходным кодом для студий разработки игр любого размера.

Если вы уже используете Unity Version Control с Unity 6, вот несколько недавних улучшений и советов, которые пригодятся вашей команде в повседневной работе.

Работайте параллельно в нескольких ветках без потери изменений с помощью Shelve и Switch

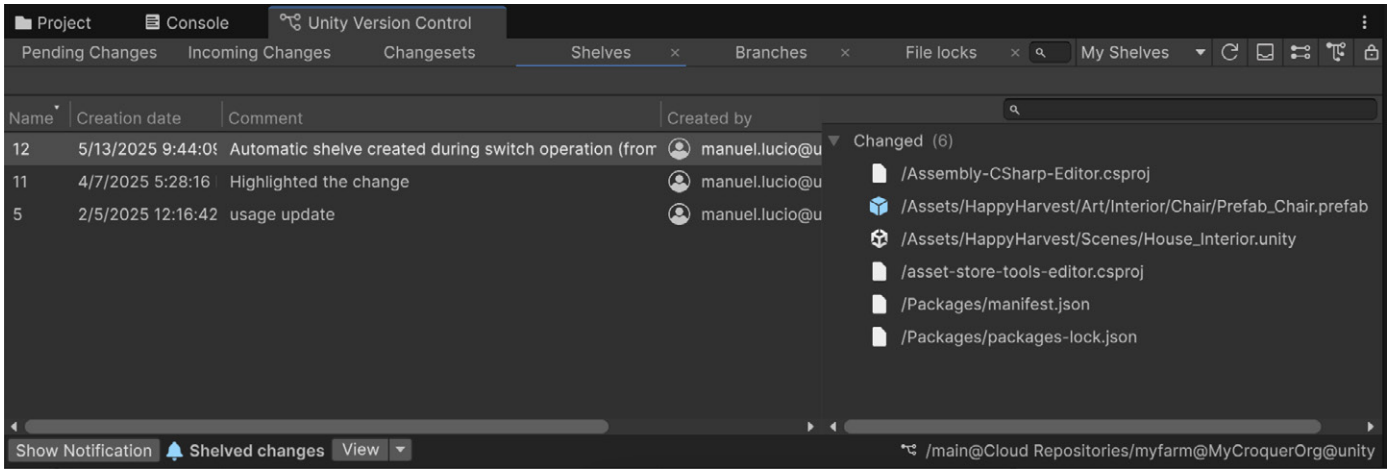
Начиная с версии пакета UVCS 2.7.1 наборы отложенных изменений (shelvesets) позволяют временно сохранять текущие правки при переключении между ветками. Так незавершённая работа остаётся в безопасности и всегда доступна.

В окне Unity Version Control щёлкните правой кнопкой мыши по ветке, на которую хотите перейти, и выберите Switch workspace to this branch.



Окно Handle pending changes

Затем выберите, перенести изменения с собой или временно оставить их. Вариант Shelves обеспечивает большую гибкость: к изменениям можно вернуться, применить их повторно и поделиться ими с командой.

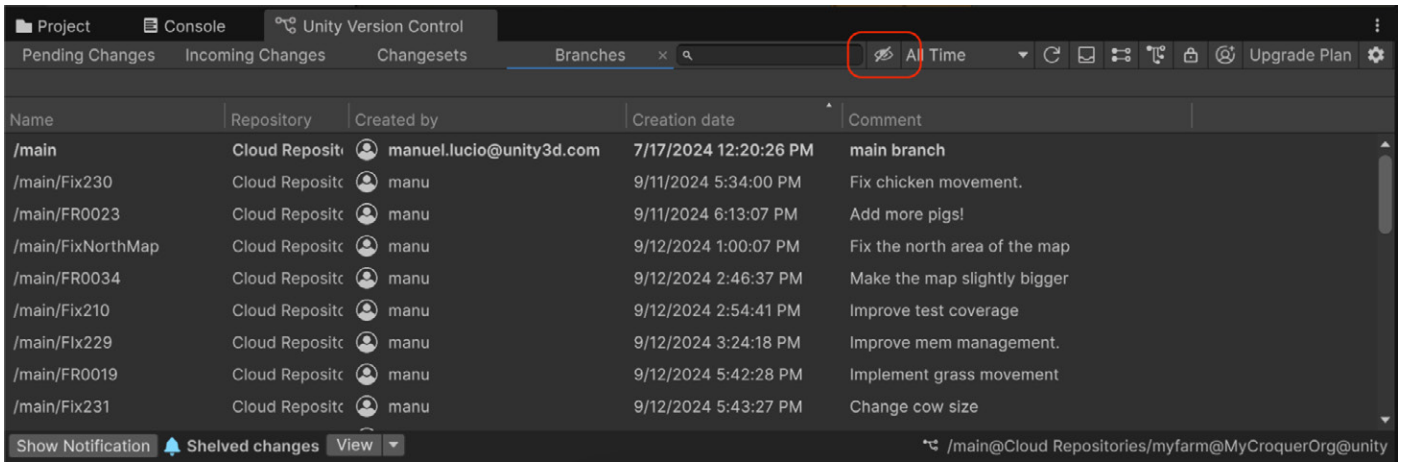


Вкладка Shelves в окне Unity Version Control

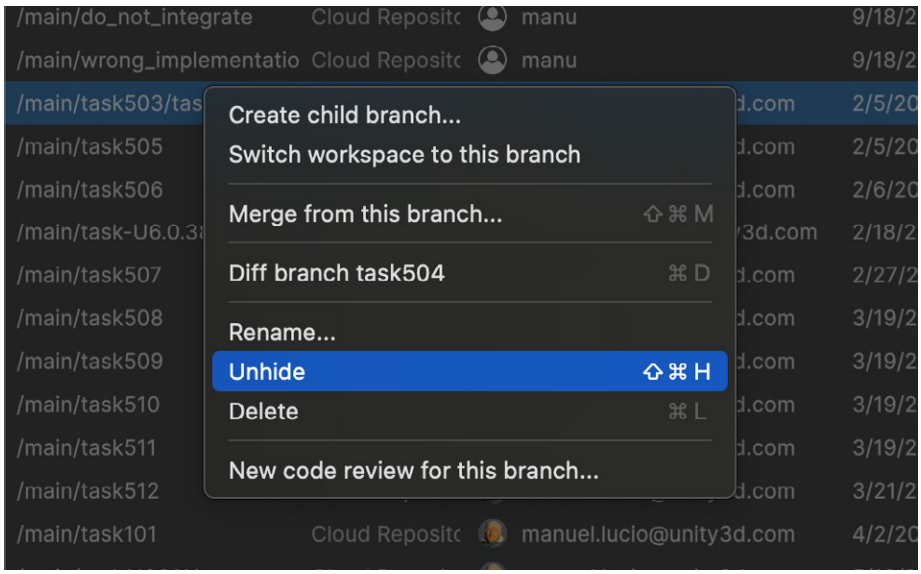
Упорядочьте список веток с помощью функции Hide/Unhide

Если представление веток проекта стало перегруженным, используйте глобальную функцию Hide/Unhide: ветки можно скрыть прямо из контекстного меню, и они станут невидимыми для всей команды на всех платформах UVCS - в плагине Unity, Unity Dashboard и настольном приложении.

Чтобы снова показать скрытые ветки, воспользуйтесь значком перечёркнутого глаза или контекстным меню списка Hidden branches (см. изображения ниже).



Значок перечёркнутого глаза в Unity Version Control

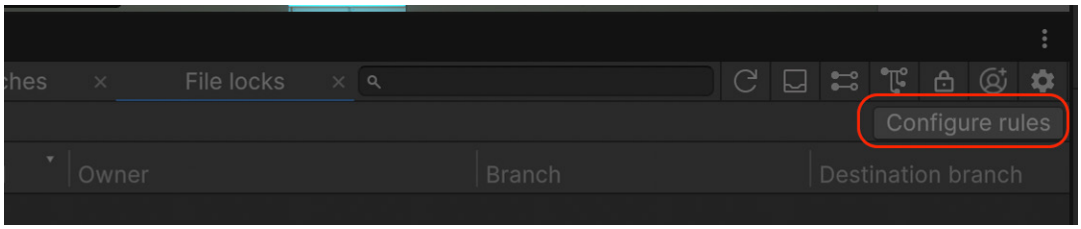


Команда Unhide в контекстном меню ветки

Защитите данные ассетов с помощью эксклюзивной блокировки

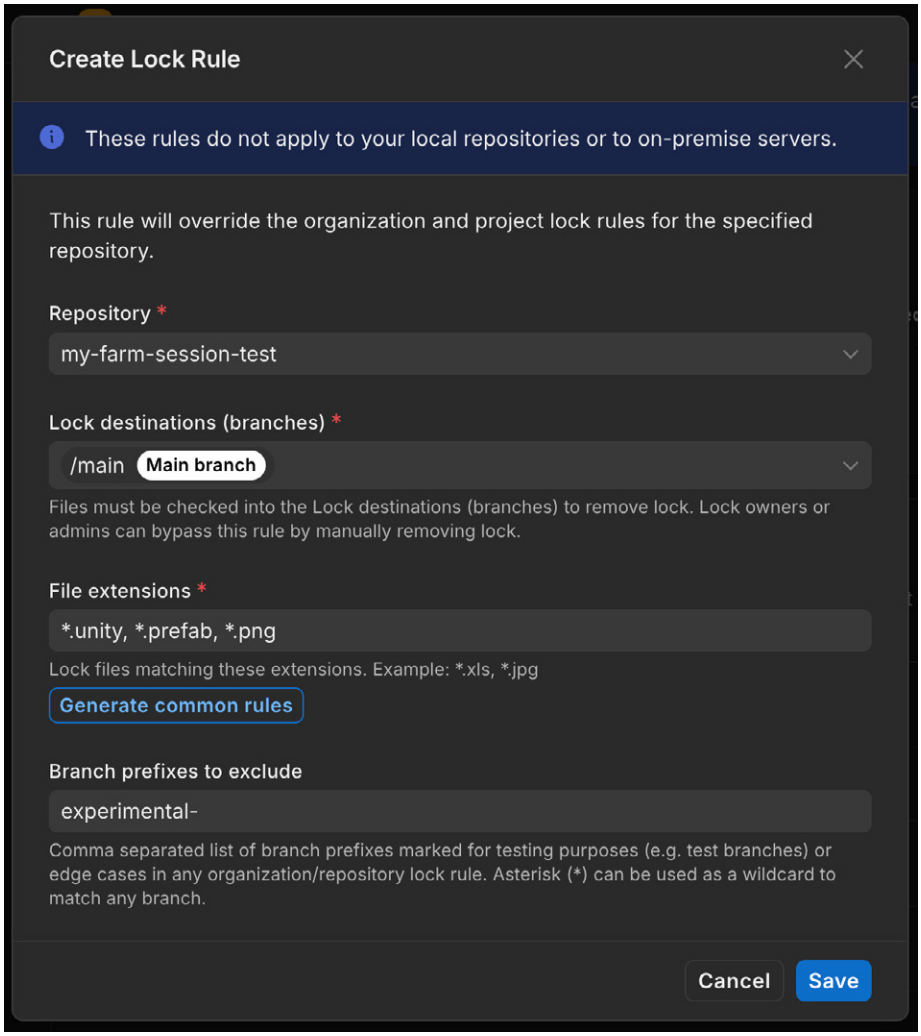
Плагин UVCS для Unity Editor позволяет заблокировать ассет, чтобы другие пользователи не могли его изменить. Это помогает команде не потерять незавершённую работу и избавляет от операции слияния для файлов, которые невозможно объединить.

Чтобы включить поддержку блокировок, откройте Configure rules на вкладке File locks.



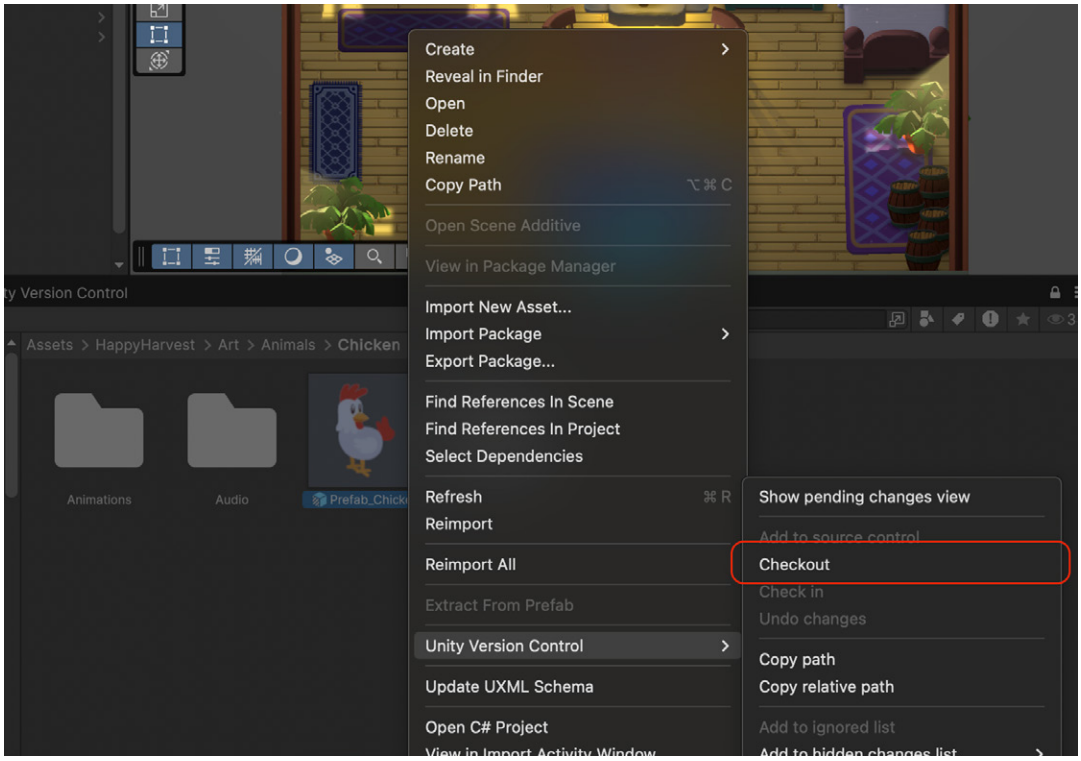
Команда Configure rules на вкладке File locks

Затем укажите репозиторий, целевую ветку - обычно основную - и расширения файлов, для которых требуется поддержка блокировок.



Окно Create Lock Rule в Unity Version Control

Чтобы избежать конфликтов, перед внесением изменений обязательно выполняйте Checkout для ассетов.



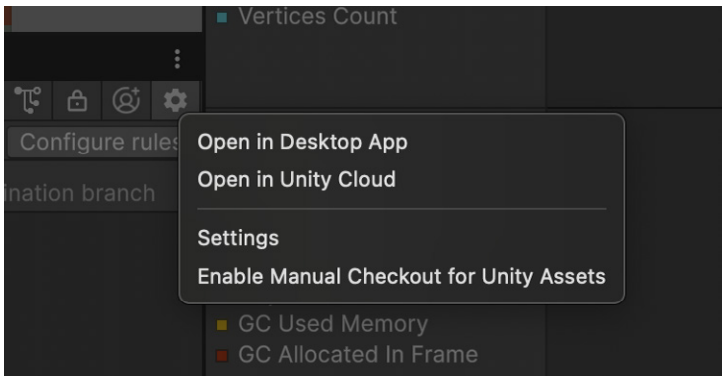
Команда Checkout в контекстном меню Unity Version Control

Подробнее о системе интеллектуальных блокировок можно узнать на странице Smart Locks.

Защитите изменения с помощью Enable manual checkout for Unity Assets

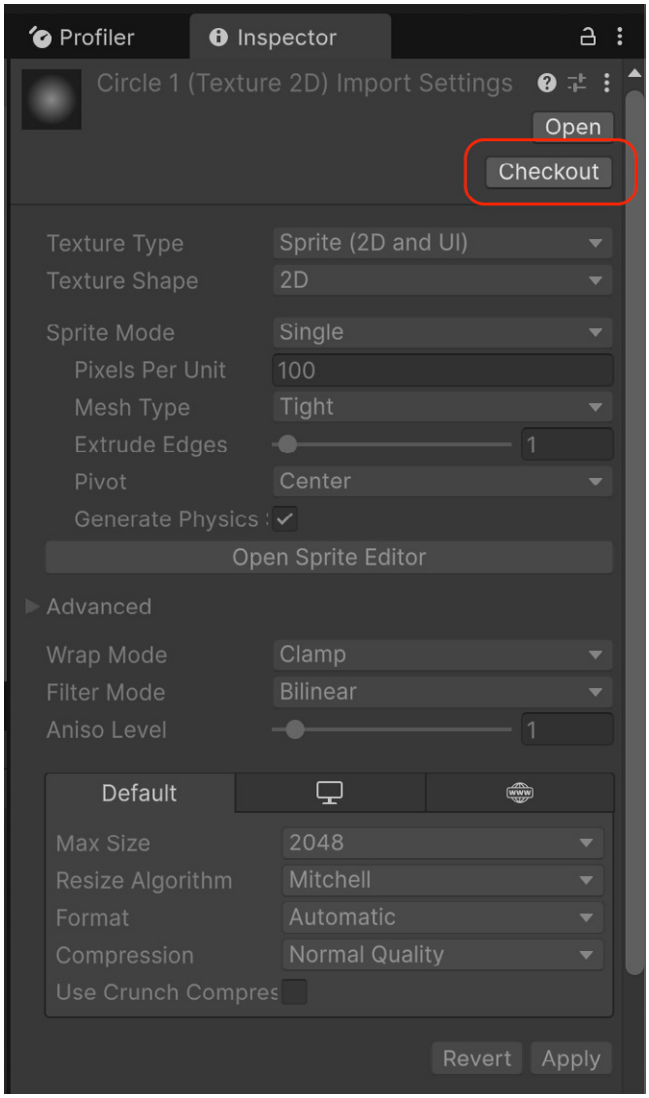
При работе с ассетами, не подлежащими слиянию, плагин UVCS для Unity автоматически выполняет Checkout файла после его изменения и сохранения. При таком порядке работы другой пользователь может успеть заблокировать тот же файл, не получив ваших изменений.

Чтобы этого избежать, откройте настройки плагина UVCS, щёлкнув значок шестерёнки, и выберите Enable manual checkout for Unity Assets.



Параметр Enable Manual Checkout for Unity Assets

Этот параметр делает ассет доступным только для чтения, пока для него явно не выполнен Checkout. После выбора Checkout в Inspector файл становится редактируемым, а вы получаете эксклюзивную блокировку на его изменение.



Команда Checkout в Inspector

Asset Manager

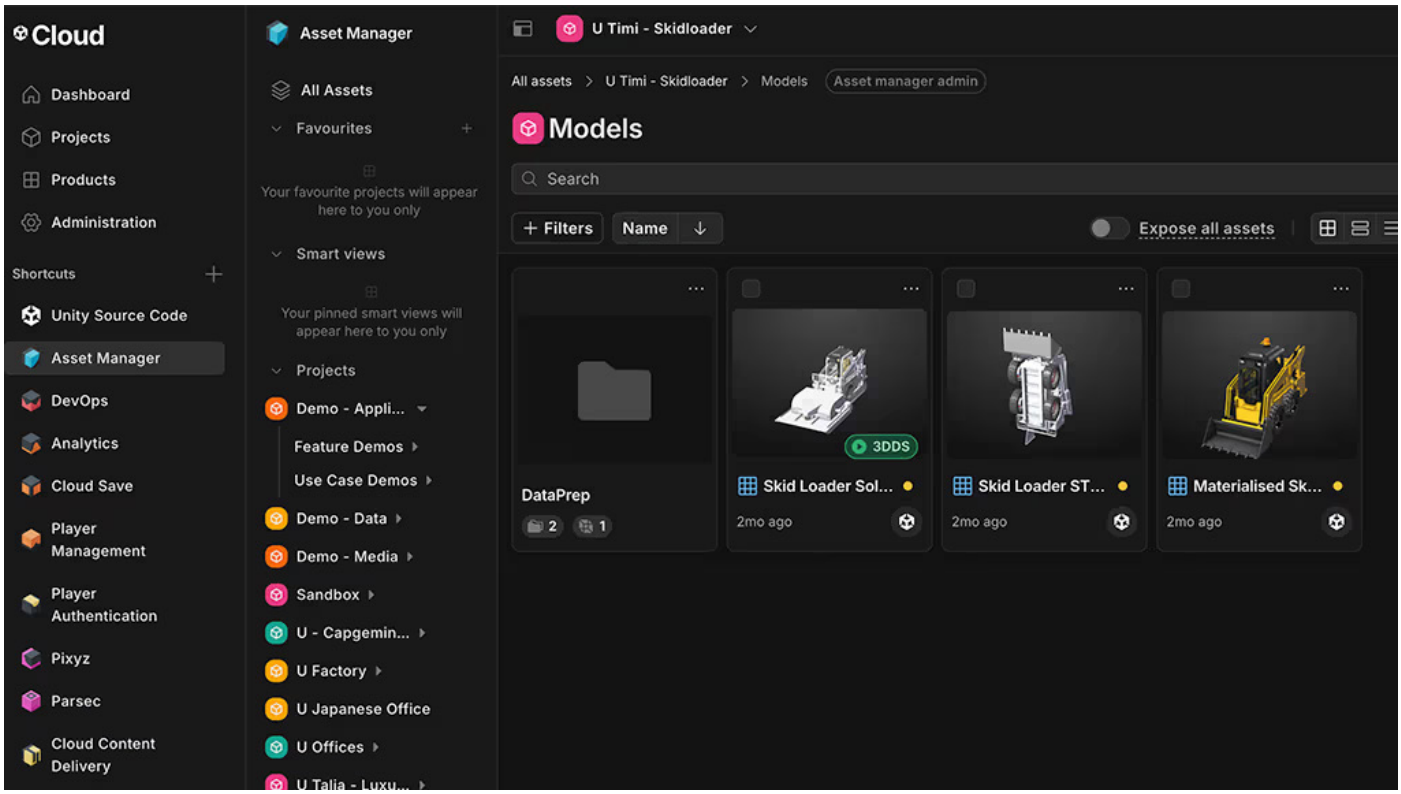
Если вам нужна система управления цифровыми ассетами - моделями, текстурами, аудиофайлами и другими материалами, - обратите внимание на Unity Asset Manager. Он поддерживает более 70 форматов файлов и помогает командам централизованно хранить, упорядочивать, находить и беспрепятственно использовать ассеты в разных проектах.

В зависимости от задач работайте с Asset Manager через веб-интерфейс или интеграцию с Editor.

Управляйте ассетами, просматривайте и упорядочивайте их в удобном браузере

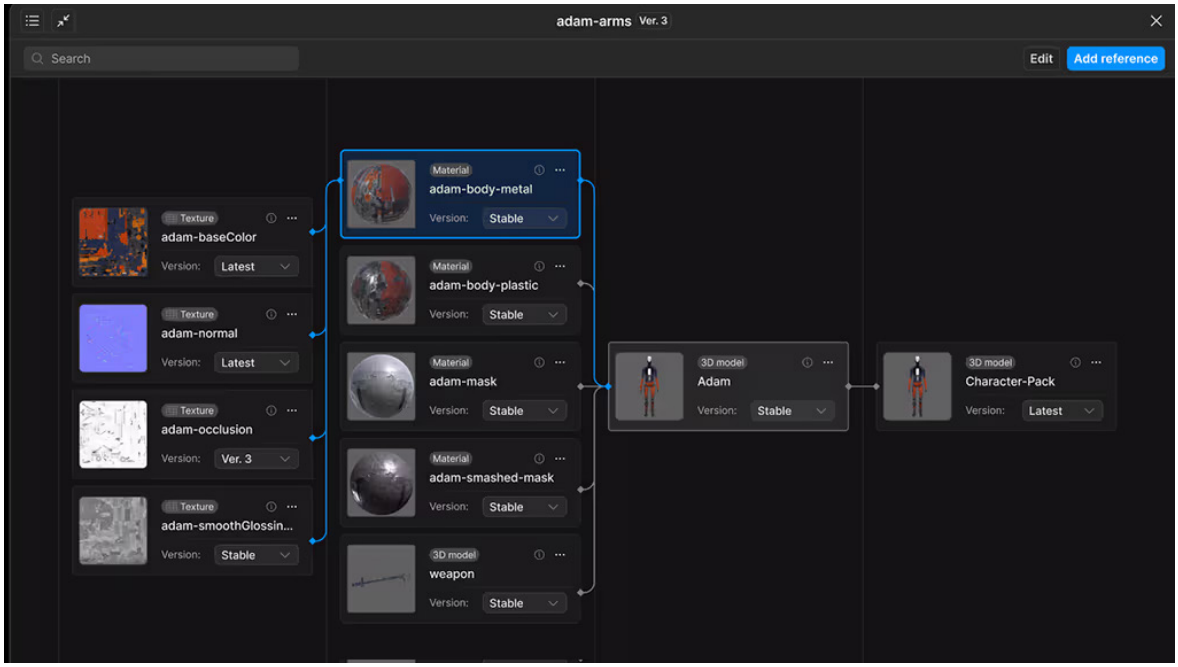
Веб-интерфейс обеспечивает быструю асинхронную совместную работу всех участников команды и заинтересованных сторон проекта. В нём можно создавать коллекции и управлять ими, перемещать ассеты между коллекциями и проектами, а также отслеживать изменения по истории версий.

Можно легко загружать один или несколько файлов, сразу добавляя их в нужные коллекции и назначая теги. Это упрощает структурирование ассетов и совместную проверку и доработку материалов. Ассеты можно дополнять пользовательскими метаданными: администраторы, владельцы и менеджеры могут определять новые поля - текстовые, логические и числовые, - а все пользователи могут добавлять теги, в том числе предложенные ИИ, чтобы ещё лучше упорядочить содержимое.



Работа с ассетами в веб-интерфейсе Asset Manager

Dependency Viewer наглядно показывает связи между ассетами: что требуется каждому ассету и какие элементы зависят от него. В Dependency Viewer доступны как входящие, так и исходящие зависимости.

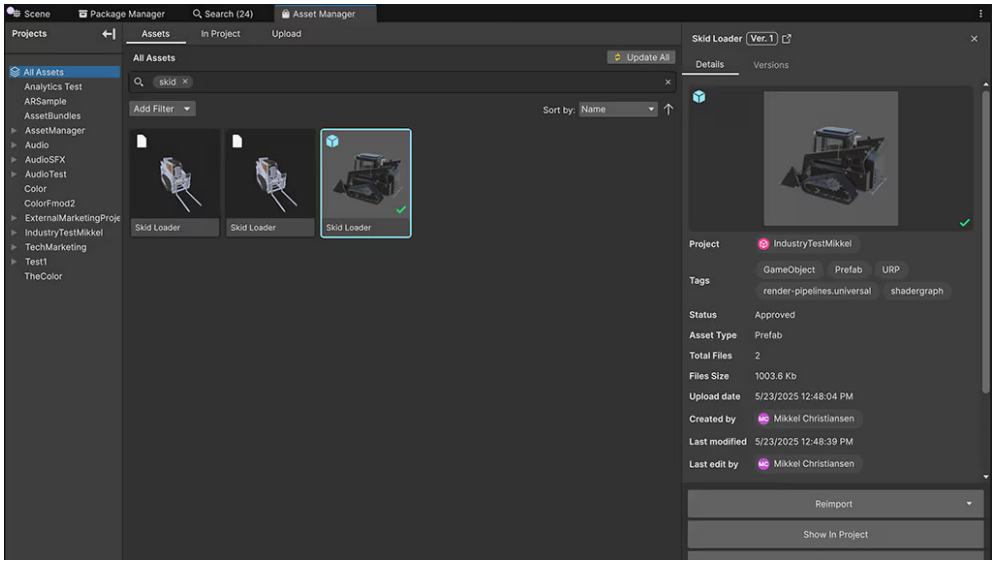


Пример зависимостей ассетов в Dependency Viewer

Управляйте ассетами в Editor без отрыва от работы

Пакет Unity Asset Manager интегрируется с Unity Editor и позволяет управлять ассетами, не прерывая рабочий процесс. После установки можно загружать ассеты в Unity Cloud и импортировать их оттуда, а также получать через поисковый интерфейс Editor доступ к ассетам всех своих проектов.

Решение, встроенное в Editor, также упрощает повторное использование префабов, скриптов и других межпроектных компонентов. Это ускоряет итерации и позволяет командам сосредоточиться на совершенствовании рабочих процессов и оптимизации.



Доступ к Asset Manager из Unity Editor

Чтобы быстро находить нужные ассеты, фильтруйте их по автору, состоянию, типу, времени обновления или состоянию импорта. Плагин автоматически определяет зависимости между ассетами. Например, если несколько префабов используют один материал, будет загружен только один его экземпляр. Это оптимизирует хранилище и предотвращает ненужное дублирование.

Подробнее о Unity Asset Manager можно узнать из следующих материалов:

- Видеоурок: краткое руководство по Asset Manager в Unity

- Unity Learn: Unity Asset Manager - краткое руководство по началу работы

- Статья: введение в способы передачи данных Asset Manager в Unity

Unity Build Automation

Unity DevOps Build Automation, ранее известный как Cloud Build, - готовое решение для непрерывной интеграции и развёртывания (CI/CD), способное выполнять и развёртывать билды в облаке. Оно позволяет чаще собирать и выпускать более качественные и инновационные версии.

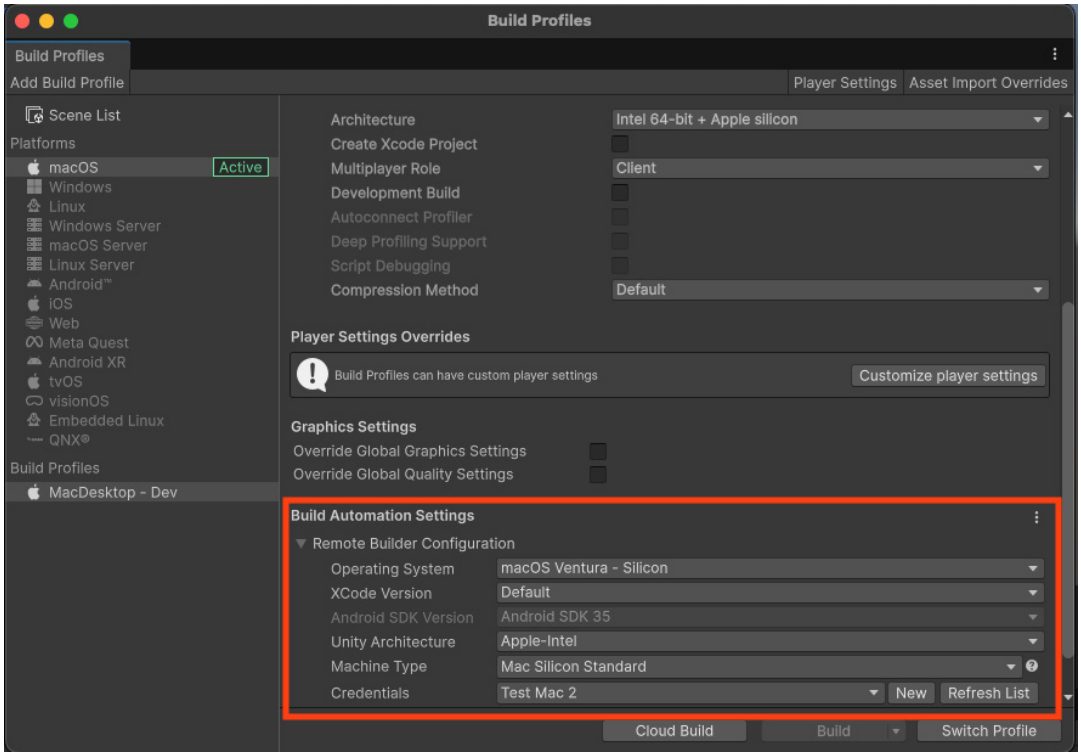
Build Automation можно за несколько минут подключить к любому репозиторию системы контроля версий и настроить на ручной запуск либо автоматическую сборку после фиксации изменений. Сервис поддерживает множество платформ, включая iOS, Android, Windows и Unity Web, поэтому не требуется обслуживать отдельную инфраструктуру сборки для каждой платформы.

Начиная с Unity 6.1 Build Automation полностью интегрирован с Editor: теперь можно запускать быстрые проверочные билды из локальных веток, используя свои настройки автоматизации сборки.

Делитесь настройками сборки и повторно используйте их в команде

Пакет Build Automation можно установить из окна Build Profiles или через Package Manager.

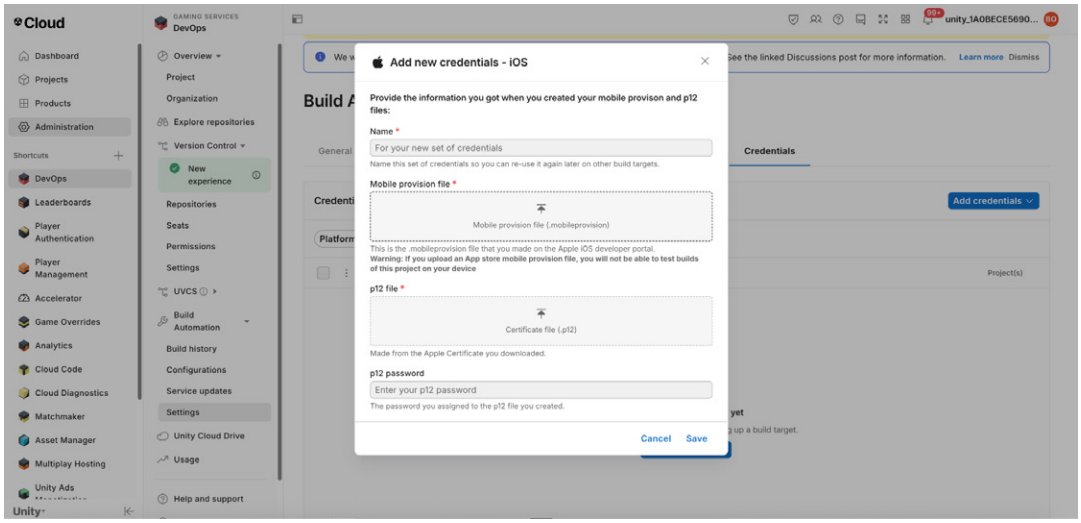
После установки пакета под выбранным профилем появится новый раздел Build Automation Settings. Эти параметры определяют, как Unity будет автоматизировать сборки.



Раздел Build Automation Settings в окне Build Profiles

Для Android, iOS и других платформ, требующих учётных данных, их необходимо настроить в Unity Dashboard. Чтобы создать новые учётные данные, нажмите New рядом с раскрывающимся списком Credentials.

Откроется новое окно в панели Build Automation, где можно создать набор учётных данных. Подробнее об учётных данных и подписывании приложений см. в документации Unity Build Automation.

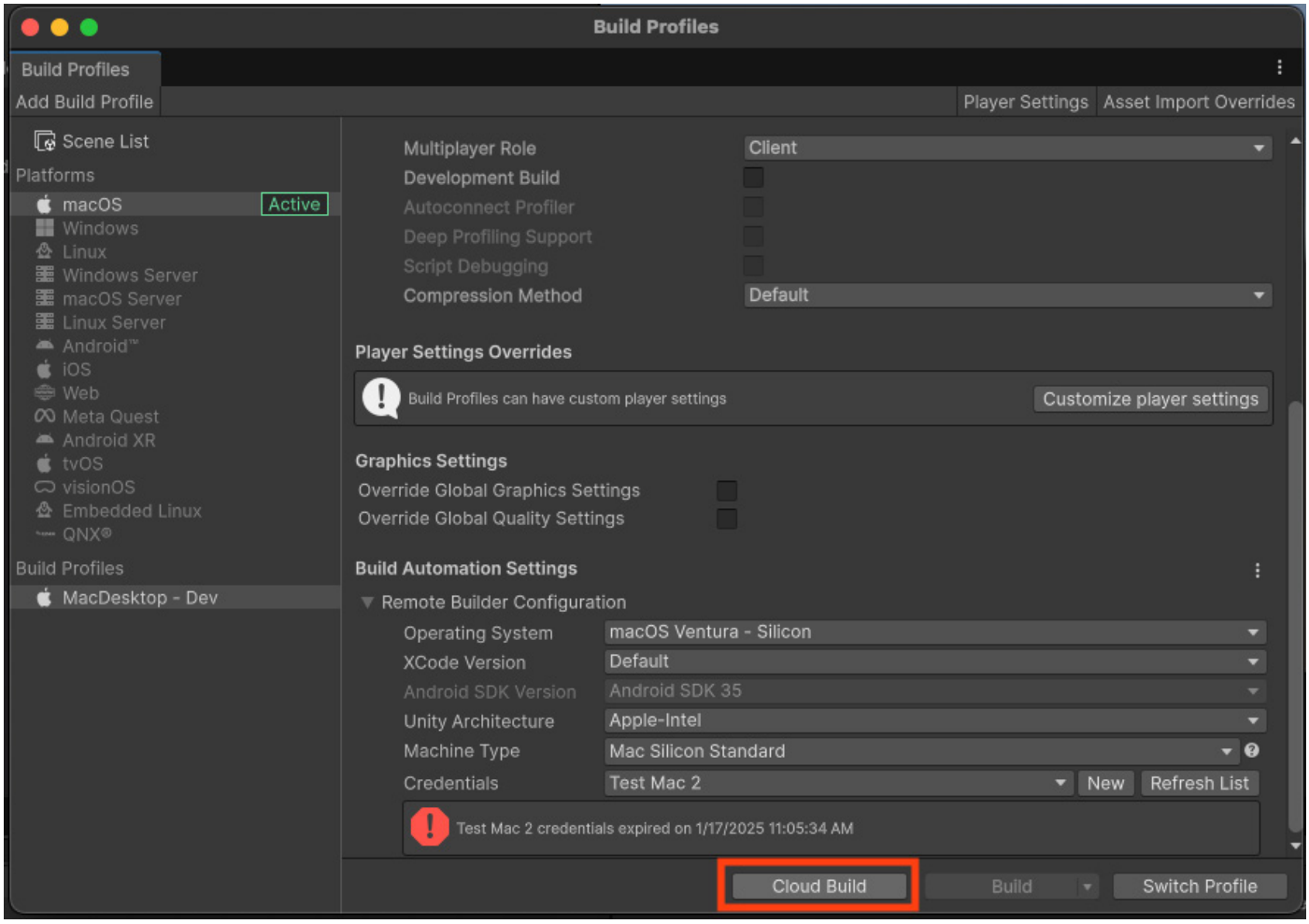


Окно настройки Credentials в Unity Dashboard

Сохранив учётные данные, вернитесь в окно Build Profiles и нажмите Refresh list. Учётные данные должны появиться в раскрывающемся списке. Параметры сборки автоматически сохраняются в профиле, поэтому их можно повторно использовать и передавать другим участникам команды.

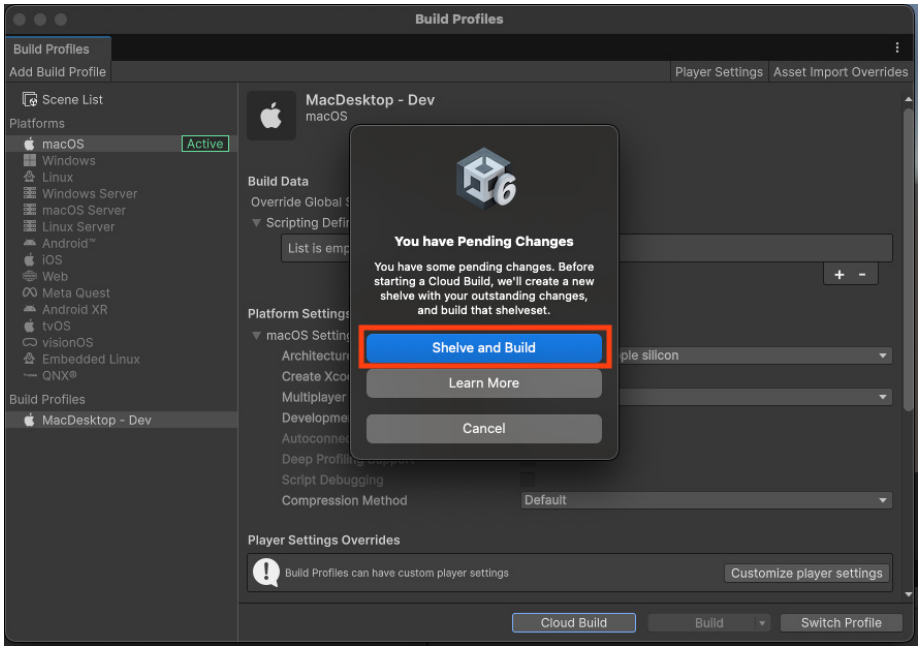
Тестируйте незавершённую работу с помощью Shelve and Build

Когда параметры настроены и всё готово к сборке, нажмите Cloud Build в профиле сборки. Unity начнёт обработку билда в облаке.



Параметры Cloud Build в профиле сборки

Если в проекте есть ожидающие изменения, появится вариант Shelve and Build. Он позволяет собрать локальные изменения, не отправляя их в основной репозиторий, что удобно для тестирования незавершённой работы.

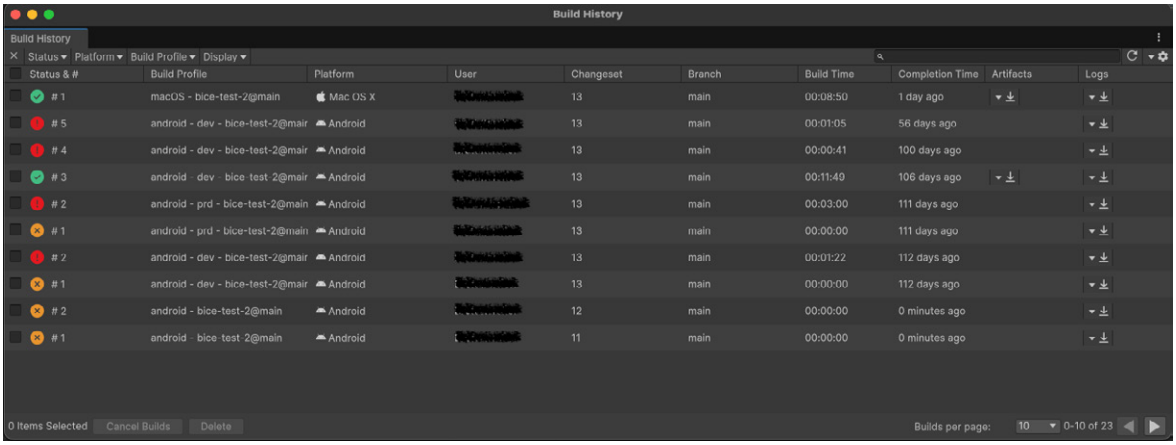


Shelve and Build для ожидающих изменений

Если ожидающих изменений нет, Unity создаст билд из последнего коммита текущей ветки.

Отслеживайте сборку, устраняйте неполадки и загружайте результаты в Build History

Сразу после начала сборки Editor откроет окно Build History, где можно отслеживать ход выполнения и состояние билда. В этом представлении доступны журналы и артефакты сборки, поэтому устранять неполадки и загружать результаты удобно.

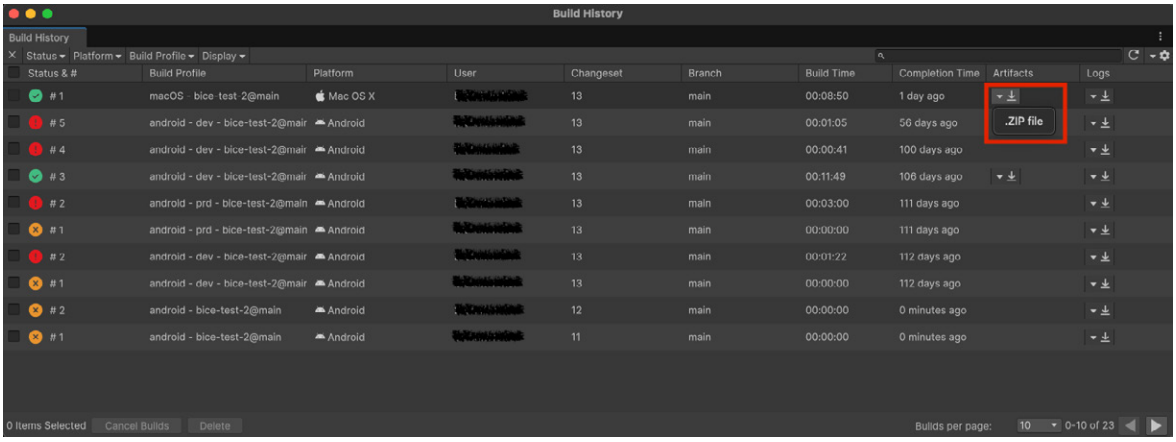


Окно Build History

Для быстрого распознавания каждое состояние имеет свою цветовую маркировку; также доступны нужные действия - например, перезапуск неудачной сборки или загрузка журнала. Полное описание всех значков состояний приведено на этой странице документации.

Делитесь с участниками команды ссылками на артефакты сборки

После успешного завершения сборки можно загрузить её артефакты или поделиться с участниками команды ссылками на них.



Загрузка артефактов успешных сборок

Для неудачных сборок можно загрузить журналы и изучить их подробнее.

Подробнее о Build Automation рассказывается в видеоуроке «Начало работы с Build Automation в Unity» и руководстве Unity Learn «Unity Build Automation: краткое руководство по началу работы».

Unity Build Server

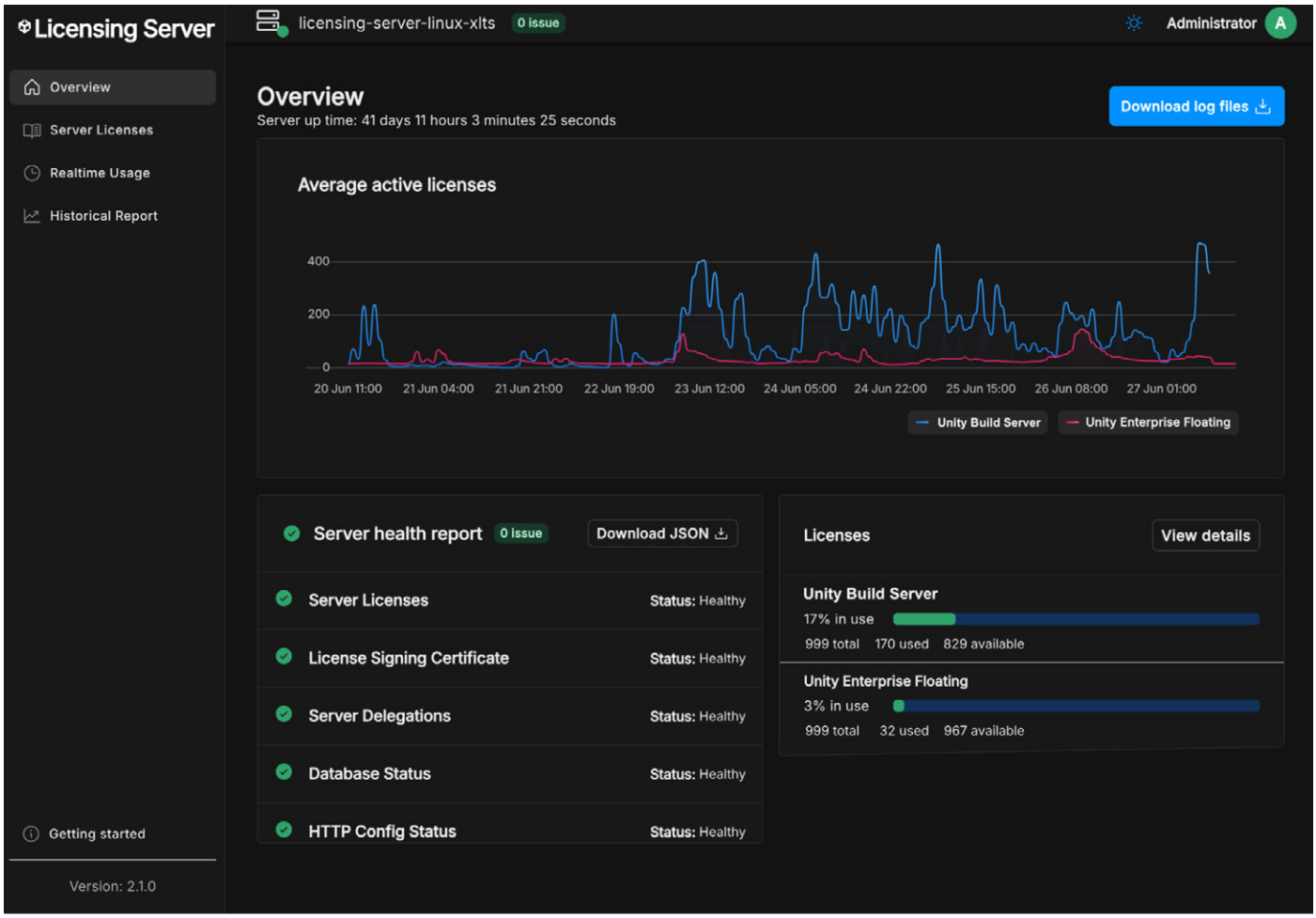
Если требуется оптимизировать и автоматизировать сборку, компиляцию и упаковку проектов Unity - особенно в крупных командах и организациях, - попробуйте Unity Build Server.

Это средство управления лицензиями предоставляет плавающие лицензии на сборку для Unity: несколько машин могут выполнять сборку без отдельной полной лицензии Unity Editor для каждой из них. Оно станет полезным дополнением к автоматизированному конвейеру сборки и средам непрерывной интеграции (CI).

Чтобы начать работу с Unity Build Server, разверните Unity Licensing Server на центральном сервере своей сети и добавьте на него лицензии Unity Build Server. Затем настройте каждую машину сборки так, чтобы при необходимости собрать проект она запрашивала лицензию у Build Server.

Unity Licensing Server, используемый в решении Build Server, может одновременно обслуживать на одном сервере несколько типов подписки. Этот же сервер можно использовать для выдачи разработчикам плавающих лицензий по подписке Unity Enterprise Floating или Unity Industry Floating. Пример показан на изображении ниже.

Требования к Unity Build Server, порядок настройки и начало работы описаны в документации Unity Licensing Server.



Две активные подписки: Unity Build Server (для машин сборки) и Unity Enterprise Floating (для разработчиков).

Демонстрационные проекты

В этой подборке есть проекты с игровыми интерфейсами, демонстрирующими рабочие процессы UI Toolkit и UI Builder, примеры различных паттернов проектирования и архитектуры проекта, а также образцы возможностей двумерного освещения и визуальных эффектов Universal Render Pipeline (URP) в Unity 6.

- Happy Harvest - демонстрационный 2D-проект

- Gem Hunter Match - демонстрационный 2D-проект

- Dragon Crashers - демонстрационный проект UI Toolkit

- QuizU - демонстрационный проект UI Toolkit

— Paddle Game ScriptableObjects

- Совершенствуйте код с помощью паттернов проектирования и SOLID
- Руководство по стилю кода C#

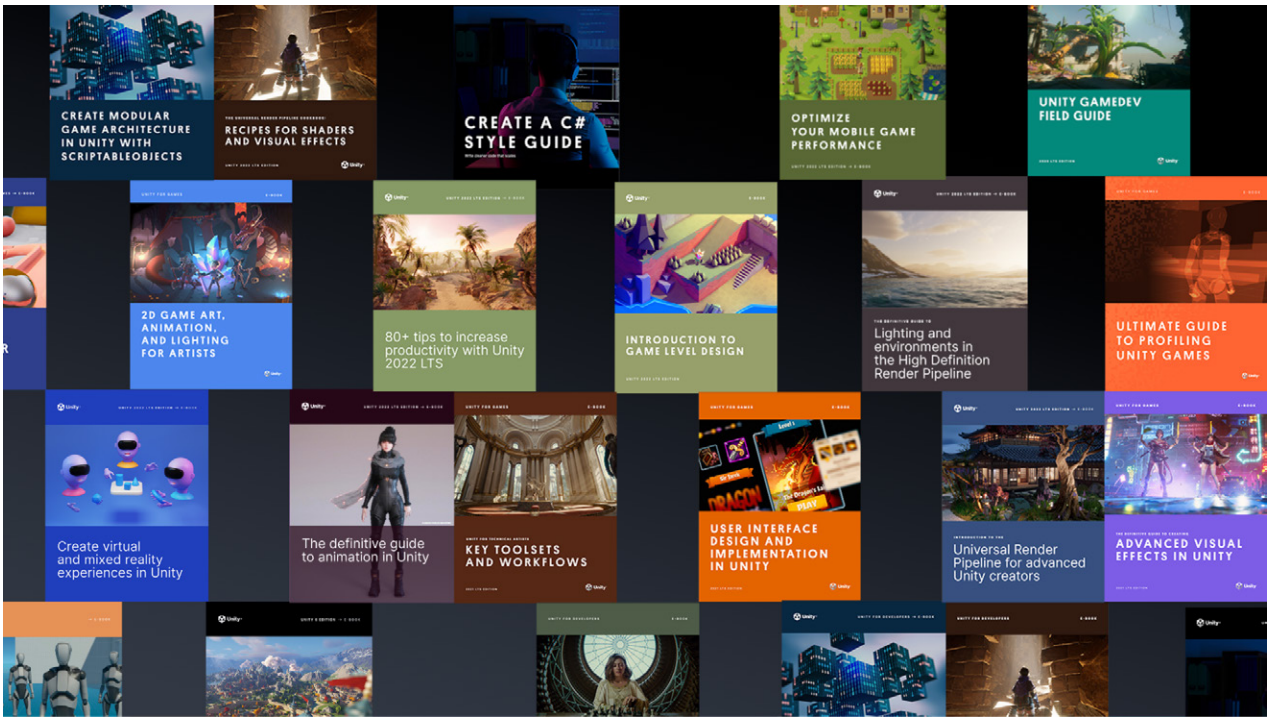
— Fantasy Kingdom

- Примеры Shader Graph
- Демонстрационное содержимое VFX Learning Templates
- Демонстрационный 3D-проект URP
- Демонстрационная сцена HDRP

— HDRP Time Ghost

- Шаблон многопользовательского проекта VR
- Шаблон многопользовательского проекта MR
- Примеры UGS

Материалы для всех пользователей Unity



В центре рекомендаций Unity можно загрузить множество других электронных книг для опытных разработчиков и авторов контента Unity. Выберите любое из более чем 30 руководств, подготовленных отраслевыми экспертами, инженерами Unity и техническими художниками: в них собраны рекомендации по эффективной разработке игр с использованием инструментов и систем Unity.

Советы, рекомендации и новости также доступны в Unity Blog, на форумах сообщества Unity, в Unity Learn и по хэштегу #unitytips.


Unity®