2. Идеальное время для одиночек: движки

Начнем с хороших новостей: вам больше не нужно изучать Assembler, чтобы сделать игру. Более того, можно вообще не знать никаких языков программирования и не получать образования программиста. Я, например, так и не удосужился этого сделать. По образованию я психолог.

Для разработки видеоигр в наши дни используются «игровые движки», коих развелось великое множество, но упоминания достойны единицы.

Игровой движок представляет собой программное обеспечение, запуск которого откроет вам такие возможности, которые не снились ни авторам Super Mario, ни авторам Final Fantasy. Представляете — в наши дни с помощью абсолютно любого современного движка можно крутить спрайты! Разработчики игр на NES и Sega MD о таком и мечтать не могли! А знаете, в каких современных движках сохранилось ограничение на вывод всего лишь 64 цветов на экран? Ни в каких! Такого ограничения больше не существует! Вы способны использовать все цвета, которые могут отобразить современные мониторы, а также сопроводить все это дело любыми звуками, которые вам приглянутся, — достаточно просто закинуть .wav или .mp3 файл в игру и не заморачиваться больше о пяти доступных каналах. Вам не нужно собирать спрайты из квадратиков 8 × 8 — вы можете нарисовать спрайт абсолютно любого размера и в любом соотношении сторон.

Перечисление таких возможностей в качестве «удивительных» может звучать саркастично и глумливо, но я отнюдь не ставлю перед собой цель высмеять инструменты прошлого. Я лишь хочу подчеркнуть, насколько доступнее и проще стала разработка видеоигр. Final Fantasy была создана, как уже говорилось, усилиями четырех человек, три с половиной из которых занимались тем, что за вас сейчас сделают движки и программы. Больше не нужно писать рендеры, больше не нужно пытаться уместить всю графику игры в смехотворные 8 Кб памяти. Шансов создать великолепное произведение у нас сейчас гораздо больше, чем было в свое время у Миямото и Сакагучи. Пока эти ребята двигались на ржавых телегах, мы сейчас будем выбирать свой истребитель среди десятка игровых движков.

«Лучшего» движка не существует. Каждый из них хорош по-своему. Потому сначала стоит хотя бы в общих чертах определиться с тем, что собой будет представлять ваш будущий проект. Будете ли вы использовать двухмерную или трехмерную графику? Планируете ли вы делать шутер, платформер или визуальную новеллу? Говорят, что ответов на эти вопросы хватит, чтобы выбрать себе игровой движок. На самом деле их хватит, чтобы лишь присмотреться к движкам, а самый главный вопрос мы зададим себе, когда поймем, какие движки вообще существуют и чем они отличаются друг от друга.

Первый актуальный движок, на который, возможно, упадет ваш взор, — это Unity. С помощью этого инструмента были созданы такие игры, как Genshin Impact, Rust и Cuphead. Он бесплатен для независимых разработчиков и, что важно, подходит для создания как двухмерных, так и трехмерных игр. Качество графики, которое вы получите, полностью зависит от вашего упорства — Unity ничем не уступает другим движкам по возможностям выдавать детализированное изображение. Но достоинства движка играют не такую важную роль, как ответ на вопрос «а может ли в нем разобраться разработчик, не обремененный лишними знаниями о программировании?».

Unity поддерживает C#, и знание C# — огромный плюс, который ускорит обучение этому изумительному инструменту. По сравнению с Assembler, на котором создавались шедевры ушедших эпох, C# является весьма высокоуровневым языком, особенно с учетом то-

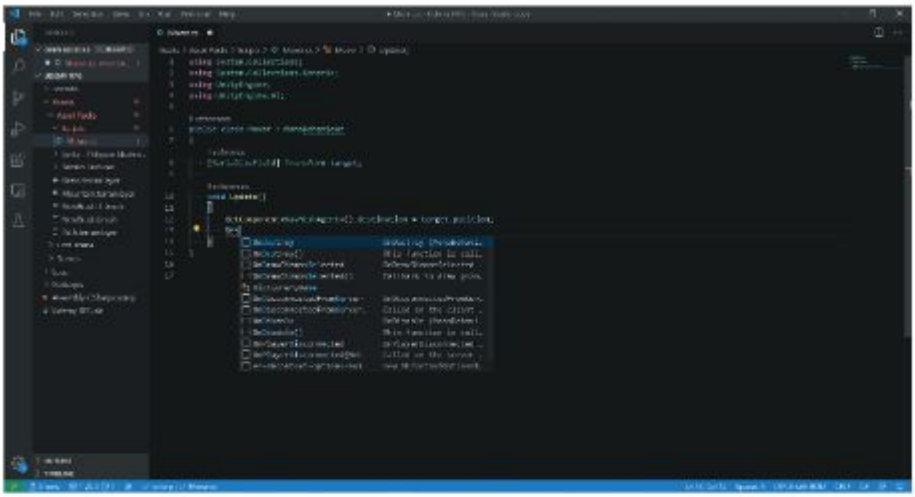

Рис. 3
Интерфейс Unity

го, что сам движок постоянно пытается угадать, что мы хотим написать, и пестрит красочными подчеркиваниями, автоисправлениями и предложениями по «правописанию» (рис. 3). Тем не менее в наши дни даже C# кажется суровым инструментом высоколобых программистов, в то время как те, кто не разбирается в C#, могут глянуть в сторону «визуального программирования» (рис. 4).

«Визуальное программирование» представляет собой не тот изнурительный процесс, в ходе которого вы пишете сложный код,

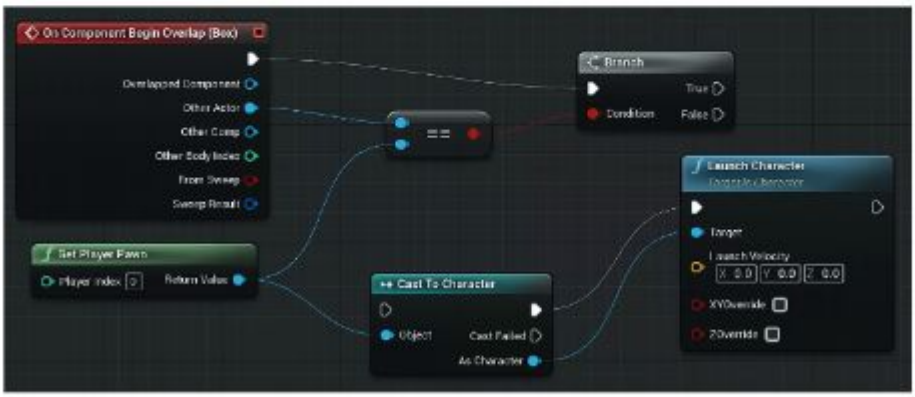

Рис. 4
Система Blue Prints в Unreal Engine

используя выражения по памяти, а потом ищете, где вы пропустили закрывающую скобку или очередную маленькую запятую. «Визуальное программирование», несмотря на красивое и сложное название, являет собой перетягивание окошек и натягивание стрелочек. Для освоения «визуального программирования» не нужны знания, которые придется получать несколько лет в университете: чтобы совладать с этим «макаронным монстром», достаточно научиться понимать логику движка. Используя свойственный движку «метод мышления», вы будете «объяснять» ему, как должна работать ваша игра. Даже запоминать никаких выражений не придется: при создании «события» вы всегда будете видеть список возможных условий.

Я — не программист. C+, C# и Python для меня являются трудноотличимыми друг от друга иероглифами. Тем не менее моим основным заработком является разработка видеоигр, и три выпущенные игры сейчас полностью обеспечивают мое существование. Удалось мне это благодаря «визуальному программированию», которое легко освоить, даже не обладая техническим складом ума.

Мне часто приходится сталкиваться с критикой моего убеждения, что «визуальное программирование» — это легко и просто, а мои слова принижают таланты людей, которые создают игры, используя тот же инструмент, что использую я. Все, что я могу порекомендовать критикам, — это почитать еще раз о том, в каких условиях и какими инструментами создавалась Final Fantasy, и тогда сразу станет понятно, чья работа представляла собой вечный поиск оригинальных решений, а чья — больше напоминает прогулку по парку в теплый летний день.

https://assetstore.unity.com/tools/visual-scripting

Рис. 5

Не нужно выстраивать вокруг себя барьер из убеждений, что вы занимаетесь сложной и изматывающей работой. Напротив, чем сильнее ваша «работа» будет ассоциироваться у вас с чем-то легким и приятным, тем больше вероятность, что и утомляться вы будете меньше. Если вы научились получать от работы в движке удовольствие — возрастет тяга к тому, чтобы сесть наконец-то за рабочее место и начать делать игры.

К психологическим барьерам мы еще вернемся — других же барьеров перед нами на самом деле не выстроено. Время снова поговорить про Unity, который гордо выставлял грудь вперед на протяжении нескольких абзацев текста, а сейчас делает огромный шаг назад. Хоть инструменты для визуального программирования на Unity и существуют (один из них называется **Bolt**), они не являют собой основной способ взаимодействия с движком и развиваются скорее «параллельно ему». Помимо Bolt есть множество других надстроек на Unity, с которыми можно ознакомиться здесь по ссылке на *рис. 5*.

Каждый из них формирует внутри движка свою собственную среду для создания игр определенного плана: платформеров, новелл, шутеров. Помимо удобного инструментария эти надстройки «награждают» разработчика неповоротливостью и неспособностью использовать функционал Unity на полную мощь, вынуждая творца все-таки прибегнуть к изучению хотя бы каких-то аспектов C# в случае, когда его воображение выйдет за рамки, которые создают подобные надстройки.

Вторым титаном среди игровых движков выступает **Unreal Engine**, блистающий своей технологичностью и всегда словно опережающий свое время. На нем были созданы Fortnite, PUBG и Hellblade. Распространяется этот мощный и серьезный движок бесплатно, но в случае если ваши доходы превысят определенное значение, команда Epic Games попросит вас оплатить Royalty — процент со своего дохода. Этим «определенным значением» сейчас выступает миллион долларов, и я искренне буду рад за тех, кому придется платить Royalty, — ведь это означает, что ваша игра стала чрезвычайно успешна.

Я не побоюсь утверждать, что именно Unreal Engine стоит за популяризацией визуального программирования: его система Blueprints уже интегрирована в движок и доступна для понимания каждому, вне зависимости от навыков и склада ума. Epic Games щедро предоставляет своим пользователям доступ к куче исходников, которые можно использовать как обучающие материалы. Например, если вы собрались делать многопользовательскую игру, то исходник **Lyra** для Unreal Engine 5 сможет стать неплохой основой для воплощения в жизнь такой, казалось бы, трудной задачи.

Кругом витает убеждение, что использование любой технологии может стать ошибкой, если технология эта применена не в той отрасли. Так, например, создание 2D-игры на Unreal Engine для многих не выглядит разумным решением: этот могучий инструмент абсолютно не предназначен для работы с плоскими картинками. Вы же не будете использовать самолет для того, чтобы добраться в «магазин-через-дорогу», верно? Ровно как и использовать велосипед для того, чтобы перебраться на другой континент, вроде бы неразумно, но именно так для многих выглядит попытка создать нечто высокотехнологичное на движках **RPGMaker**, **Construct**, **Gamemaker** или **Ren-py**. Все они предназначены для работы только с 2D-графикой, а Ren-py так и вовсе разработан только для создания визуальных новелл.

3. Движки и ваши личностные особенности

Я только что рассказал про два могущественных движка, которые позволяют воплотить в жизнь любые по сложности идеи, но тем не менее на Unity и Unreal Engine создается лишь около половины игр, выпускаемых в Steam. Некоторые крупные AAA-студии продолжают использовать свои собственные закрытые движки, чтобы меньше зависеть от сторонних компаний, и к этим ребятам вопросов у нас не имеется. Но что же движет небольшими студиями и соло-разработчиками, которые выбирают себе в качестве основного инструмента другие, куда менее популярные и продвинутые продукты?

Unreal Engine и Unity, по сути, не сложнее в освоении, чем другие «мелкие» и «неповоротливые» движки. С одной стороны, научиться на них работать даже проще: у Unity есть официальный бесплатный курс для начинающих разработчиков, а уж количество обучающих материалов по Unreal Engine зашкаливает, в то время как найти толковый обучающий курс по какому-нибудь Construct — задача непосильная.

Люди, выбравшие инструментом для воплощения своих идей весьма примитивные программы, на мой взгляд, опровергают популярное убеждение, упомянутое мною выше: якобы самое важное — чтобы инструмент использовался по назначению.

Но главное в выборе движка — это ваши личностные особенности. От них-то и надо отталкиваться.

Если вам будет некомфортно работать на выбранном движке, то вы не будете на нем работать. При разработке игры в одиночку нам ничто не мешает опустить руки и бросить свои светлые начинания. Нам важно создать условия, в которых вероятность устать от создания игр будет минимальной.

Ключевым нюансом в выборе движка является то, насколько вам приятно и удобно с ним работать. Уделите по одному вечеру каждому из движков, выполните в них по одному уроку и определитесь с тем, где вам было приятнее работать.

Даже если вы собрались делать 2D-игру, но вас привлекают эстетика «высокотехнологичности», сияющая новизна и чувство причастности к чему-то очень продвинутому — выбирайте Unreal Engine. Если же вам неприятно работать в Unreal Engine и он кажется вам излишне перегруженным, то какой бы проект вы ни делали с помощью этого инструмента — он навсегда останется недоделанным.

Если у вас есть планы найти работу в среде разработчиков, делать мобильные приложения или устроиться в игровую студию, то Unity на данный момент является движком, навык владения которым быстрее всех поможет вам найти работу. Если же вы все время спотыкаетесь о C#, чувствуете, как написание кода вас тормозит и мешает воплощать идеи в жизнь, а использование 3D для имитации 2D кажется вам глупостью, то у вас не получится раз за разом возвращаться к интерфейсу Unity и создавать свой продукт.

Construct — движок для работы только с 2D-графикой. Именно он — мой личный выбор. Construct полностью основан на визуальном программировании и создает среду, в которой работа с неким подобием кода протекает невероятно быстро. Я человек абсолютно не технического склада ума. Меня не пугает, что я не могу заглянуть «под капот» движка и до конца понять все процессы, которые скрываются за аккуратными «блоками» с условиями. Мне это просто неинтересно. Я могу уделить больше внимания спрайтам и визуальной составляющей моей игры. Именно внешний вид создаваемых мной продуктов для меня является важнейшим аспектом: я одер-

Рис. 6

Логика поведений в Construct. Слева — условие (пуля коснулась игрока), справа — событие (игрок уничтожен)

жим контролем над каждым спрайтом, над каждой тенью и каждым эффектом, оттого предпочитаю классическую анимацию и полное отсутствие программных спецэффектов и постобработки в своих проектах. Такие игры, как **Hollow Knight**, **Cuphead** и **Hotline Miami**, не были созданы на Construct2, но могли бы — движок позволяет создать все то, что мы видели в этих играх (рис. 6).

Упомянутая Hotline Miami (рис. 7) была создана с помощью **Gamemaker**. Точно так же как и Construct, этот движок запирает разработчика в двух плоскостях, лишая его возможности создавать трехмерные игры. В Gamemaker используется собственный язык GML, который имеет и визуальное ответвление. GML отличается от языка Construct своей глубиной и сложностью, что делает Gamemaker инструментом, подходящим для людей с техническим

Рис. 7

Hotline Miami 2: Wrong Number. *Dennaton Games, 2015*