УДК 004.02

Увеличение продуктивности в сфере разработки игр с помощью автоматизации процессов

© В.А. Артемьев
Иркутский национальный исследовательский технический университет,
г. Иркутск, Российская Федерация

Аннотация. Переработки в сфере разработки игр не являются редкостью и считаются «нормой» для игровой индустрии. Игнорирование проблемы приводит к высокой текучке кадров и выгоранию. Часть работников навсегда отказывается участвовать в разработке игр, причиной этому становится сильный стресс и большая вероятность работы на износ. Одной из частых причин возникновения переработок является плохо организованный процесс разработки, при котором большая часть временного ресурса может быть потрачена впустую. При анализе типичных рабочих процессов, проходящих при разработке игр, были обнаружены рабочие операции, которые можно оптимизировать с помощью автоматизации, тем самым уменьшить время, требуемое на завершение работы. В результате сравнения эффективности рабочего, использующего автоматизацию одной операции, и рабочего, который делает эту операцию вручную, были сделаны выводы, заключающиеся в том, что автоматизация даже малой части рабочего процесса приводит к существенному увеличению уровня производительности. Таким образом, при возникновении в компании «кранчей» оптимизация и автоматизация текущих рабочих процессов может поспособствовать уменьшению переработок.

***Ключевые слова*:** переработки, разработка игр, автоматизация, работа

Production Gain in Game Development with Process Automation

© Vladislav A. Artemiev
Irkutsk National Research Technical University,
Irkutsk, Russian Federation

Abstract. Overtime in the field of game development is not uncommon and is considered to be the "norm" for the gaming industry. Ignoring the problem leads to high staff turnover and burnout. Some workers forever refuse to participate in the development of games; the reason for this is severe stress and a high probability of working for wear. One of the common causes of overtime work is a poorly organized development process, in which most of the time resource can be wasted. When analyzing typical game development workflows, work operations were found that can be optimized through automation, thereby reducing the time required to complete work. As a result of comparing the efficiency of a worker who uses the automation of a single operation and a worker who does this operation manually, it was concluded that the automation of even a small part of the workflow leads to a significant increase in the level of productivity. Thus, when crunches occur in a company, optimization and automation of current work processes can help to reduce overtime.

***Keywords*:** overtime, game development, automation, work

Время – самый важный ресурс, используемый при разработке любого вида программного обеспечения. Своевременный выпуск продукта определяет значительную долю его успеха, ведь в современном мире технологии развиваются и меняются с огромной скоростью. Задержка выхода продукта может привести к тому, что к моменту его выпуска он будет устаревшим и его могут начать переделывать, таким образом, такой продукт может стать vaporware [1], или ему придётся конкурировать с более предприимчивыми разработчиками, которые смогли выпустить похожий продукт раньше.

Разработка игр – очень долгий процесс, включающий в себя огромное количество других сфер медиа: программирование, рисование, моделирование, анимация, музыка, звуки и прочее. Каждая из этих сфер зависит от влияния времени, особенно всё, что связано с графикой и анимацией. Задержки в разработке игр нежелательны, ведь никто не хочет играть в устаревший на выходе продукт. Поэтому целью большинства команд разработчиков является выпуск продукта как можно раньше. Однако чаще всего разработчики используют самый простой и самый непродуктивный метод – увеличение количества рабочих часов в неделе. Причём

такие переработки могут даже не оплачиваться из-за того, что многие компании пытаются внушить, что разрабатывать игры – это привилегия, тем самым склоняют работника трудиться на «благо» компании [2].

Переработки являются одной из основных проблем при работе в GameDev. Почти каждая крупная компания заставляет людей перерабатывать, чтобы успеть завершить продукт в срок. Основными причинами возникновения переработок становятся плохая согласованность различных отделов и трата ресурсов на лишние действия (рутину). Чем меньше людей в команде, тем больше рутинных задач они выполняют. В итоге переработки приводят к:

1) падению производительности команды;
2) выгоранию;
3) текучке кадров [3].

Следствием переработок являются крупные проблемы при работе над проектом и ещё большие задержки.

Многие компании понимают, что у них существует проблема переработок, но почти не пытаются её решить.

При работе над Red Dead Redemption 2 сотрудники Rockstar порой работали по сто часов в неделю. Это стало известно из интервью с более чем 70 сотрудниками. Хоть Rockstar и позволила сотрудникам разговаривать с журналистами, почти все люди, которые давали интервью, пожелали остаться анонимными. Некоторые заявили, что опасаются возмездия за откровенный рассказ о своём негативном опыте в Rockstar [4].

Марчин Ивински, глава и сооснователь CD Projekt Red, называет переработки «необходимым злом», он добавляет, что «разработка игр – это тяжёлая работа, которая может разрушить вашу жизнь» [5].

Вышеперечисленные компании выпустили свои новые игры совсем недавно, что показывает, что проблема с переработками до сих пор актуальна и никуда исчезать не собирается. Почти все разработчики привыкли к текущему положению дел и не пытаются как-то изменить свои подходы к разработке. Ведь мало кто любит выходить из зоны комфорта привычного пайплайна и привыкать к новым вещам, даже если старый пайплайн не оптимален и разрушителен для работника.

Многие исследования показывают, что переработки на самом деле замедляют разработку проекта. В 1980 году Business Roundtable опубликовали отчёт «Scheduled Overtime Effect on Construction Projects», в котором утверждалось, что производительность при 60-часовой рабочей неделе ниже, чем при 40-часовой рабочей неделе.

«Если рабочий график, состоящий из 60 или более часов в неделю, продолжается более двух месяцев, совокупный эффект снижения производительности вызовет задержку даты завершения проекта сверх той, которая могла бы быть достигнута с тем же размером команды при 40-часовой неделе», – такое заключение приводится в исследовании [6].

Двадцать дополнительных часов действительно увеличивают общий объём производства, но это применимо только к началу переработок. Исследователи утверждают, что производительность упадёт очень быстро. Примерно за два месяца производительность может снизиться до такой степени, что проект был бы выполнен гораздо раньше, если бы не было переработок и все придерживались 40-часовой рабочей недели [7].

Можно решить некоторые проблемы, возникающие во время разработки и ведущие к переработкам, с помощью автоматизации процессов.

Посредством автоматизации можно вывести качественные и количественные показатели процессов на более высокий уровень, также автоматизация способствует увеличению производительности, ускорению работы, удешевлению и увеличению точности и стабильности.

При проектировании автоматизированного процесса важно учитывать основные принципы автоматизации:

1) согласованность. Все действия должны быть согласованы как взаимно, так и со входами и выходами процесса;
2) интеграция. Процесс должен встраиваться в общую среду и обеспечивать взаимодействия процесса с внешней средой;
3) независимость исполнения. Человек не должен участвовать в процессе.

Каждый элемент игры разрабатывается почти в полной независимости друг от друга, связывает их лишь дизайн-документ или другие требования тимлидов. После завершения работы над этим элементом его нужно добавить в игру. Однако не всё так просто. Как быстро интегрировать что-то в систему при отсутствии самой системы? Кто должен заниматься интеграцией: программист, работающий над кодом, геймдизайнер, непосредственно создающий игру, или

человек, создавший ассет, который нужно интегрировать? Или всё-таки необходимо создать систему, которая сама будет импортировать элементы?

На примере взаимодействия художника и программиста рассмотрим несколько возможных ситуаций, возникающих при работе над игрой, и попробуем проанализировать их продуктивность при различных методах интеграции элементов.

Предположим, что у нас есть художник, создающий модели и другой контент, который потом пойдёт в игру. Он закончил создание очередного ассета. Теперь его необходимо интегрировать в игру, редактор или что-то другое для продолжения дальнейшей разработки. Упрощённый пайплайн от создания до имплементации выглядит таким образом:

1) создаётся ассет в сторонней программе (в данном случае трёхмерная модель, которая будет использоваться в игре);

2) художник экспортирует ассет из программы с нужными параметрами и форматом файла в соответствии с требованиями заказчика или тимлидов;

3) ассет импортируется в другой инструмент (в данном случае в движок), и производится настройка нужных параметров при импорте (например, выбор скелета для модели);

4) производится проверка на ошибки (лишние кости при импорте, неправильная ориентация поверхностей и т. д.);

5) в случае обнаружения ошибок ассет отправляется на доработку;

6) ассет подготавливается для использования в игре.

Рассмотрим случай, когда импортом и имплементацией занимается программист. При таком пайплайне у имплементирующих в итоге остаётся меньше времени для того, чтобы решить свою задачу, потому что некоторое время уходит на корректный импорт, поиск ошибок и т. д. После этого ассет либо отправляется обратно художнику, либо вводится в игру. Проблема такого подхода заключается в лишней нагрузке на программиста, у которого есть другие задачи, кроме проверки и добавления очередного ассета в игру, что замедляет непосредственную разработку.

Перейдём к случаю, когда импортом занимается художник. Плюсом данного подхода является то, что при выявлении ошибки художник может быстро перейти к её непосредственному исправлению, а не ждать, пока ему отправят отчёт. Такой подход уменьшает время на создание новых ассетов и одновременно освобождает программистов от лишней работы. Но вся эта работа является рутиной, а рутина уменьшает время на выполнение другой, более важной и уникальной работы. Из-за недостатка времени на выполнение нужной работы под давлением вышестоящего менеджмента сотрудники начинают перерабатывать, чтобы успеть в срок [8].

Изучив предыдущие подходы, мы видим, что экспорт, импорт и проверка являются теми задачами, которые можно автоматизировать, тем самым сэкономить большое количество времени. В итоге у нас получится пайплайн, представленный на рисунке 1.

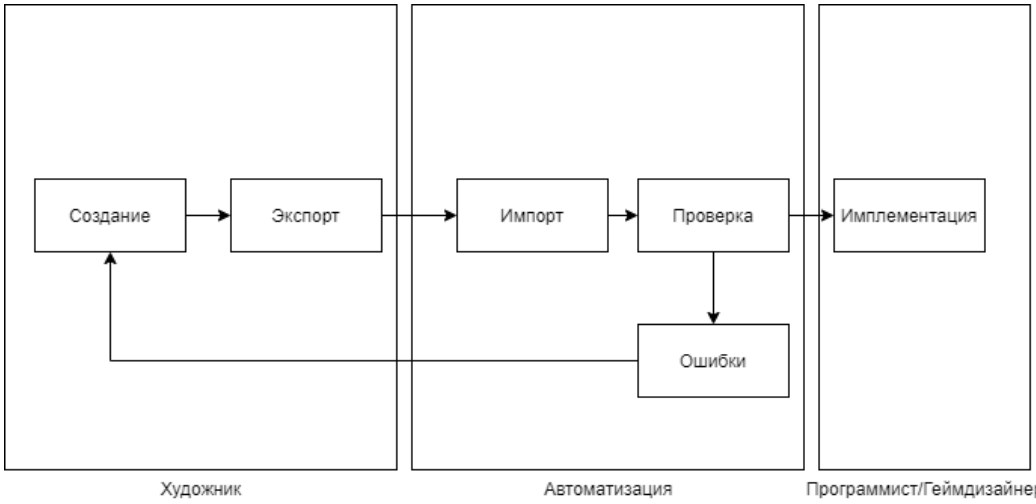

Рис. 1. Пайплайн с автоматизацией

Каждый занимается своим основным делом и не отвлекается на рутинные вещи. Художник экспортирует туда, куда нужно, программист или геймдизайнер получают

ассеты уже в готовом для имплементации виде. Однако такой подход требует строгой стандартизации экспортируемых материалов, так что человеческий фактор всё ещё имеет место, но уже не является столь значительным. Конечно, можно включить шаг экспорта в процесс автоматизации, но это требует от программиста изучения ещё одного API в программе для моделирования/создания звуков/текстурирования и т. д. Поэтому вопрос о том, включать этот шаг или нет, зависит от хода разработки. Это избавит от необходимости вручную подгонять всё под стандарт при экспорте, но увеличит время на создание процесса автоматизации.

В итоге с помощью автоматизации мы можем избавиться от:

1) лишних ручных и рутинных процессов в пайплайне;

2) необходимости переключаться между разными процессами разработки (художнику или программисту придётся запускать другую программу, кардинально отличающуюся от той, с которой они работали пять секунд назад, кроме этого, ожидание запуска этой программы увеличивает суммарное время, затраченное на импорт);

3) переработок, которые возникают вследствие того, что работники не успевают выполнить свою основную часть работы;

4) стресса (все пункты, перечисленные выше, добавляют лишний стресс, что в свою очередь уменьшает производительность).

Такую автоматизацию можно применить и к другим элементам игры. Добавление локализации, музыки, эффектов, создание и установка билдов, тестирование и т. д.

Таким образом, актуальность исследования обусловлена необходимостью разработки инструментария, который позволит уменьшить время, требующееся на взаимодействие между разными программами, и тем самым увеличить продуктивность работников.

Новизна работы заключается в том, что в данный момент такие программы, называемые Bridge, крайне редкие, узкоспециализированные и обладают небольшим функционалом (например, переносят одну простую модель, даже без материалов), такой программы недостаточно для существенного увеличения производительности.

Продолжим рассматривать пример взаимодействия 3D-художников с программистами. Реализацию импорта лучше всего писать на Python, так как большинство таких программ имеет документацию по Python API. В данном примере скрипт был написан с использованием Unreal Engine 4 Python API.

Благодаря одному такому скрипту можно избавиться от большинства рутинной работы, потому что с использованием API движка можно импортировать и настраивать любой тип принимаемого файла [9], а также создавать необходимые классы для использования этих файлов, то есть можно объединить процесс импорта и имплементации в одну команду.

Требования к импортируемым и создаваемым файлам будут отличаться от проекта к проекту, поэтому такой скрипт нельзя назвать универсальным, его придётся модифицировать или создавать заново. Но он всё равно позволит ускорить определённые этапы разработки.

Допустим, художник предоставил готовую модель с текстурами и другими дополнительными файлами. Теперь все эти файлы необходимо импортировать в движок. Но при импорте текстуры не наложатся автоматически на модель, для этого нужно создавать отдельный материал, настраивать там каждую текстуру и только после всего этого применять её на модель. Однако и модель тоже нужно правильно импортировать, ведь у неё в файле может быть записана информация о скелете и анимациях, которая при неправильном импорте может потеряться.

После операций импорта модель требуется подготовить для использования в игре. Обычно модель привязывают к объекту какого-то заранее созданного класса и потом просто меняют там параметры под необходимые. Этот шаг тоже можно автоматизировать, так как API движка позволяет создавать объекты не только стандартных для движка классов, но и созданных во время работы над проектом.

Через несколько секунд после запуска скрипта мы имеем готовый к использованию в проекте ассет.

С применением всей этой информации был написан скрипт, который делал следующее:

1) импортировал из папки текстуры и модель;

2) модифицировал материал модели с необходимыми текстурами;

3) создавал простого актора;

4) создавал дата ассет кастомного типа с нужными полями для ввода данных.

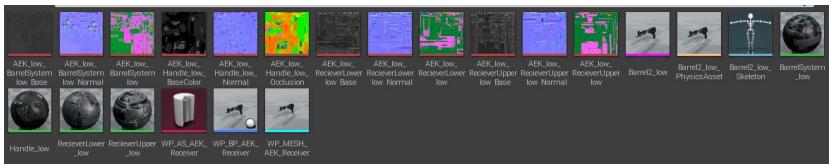

Рис. 2. Итоговый набор файлов после работы скрипта

В течение нескольких секунд после запуска скрипта можно наблюдать готовый к использованию в проекте ассет, представленный из файлов, которые можно посмотреть на рисунке 2. Для сравнения, подготовка каждой модели вручную занимает около 15 минут. Казалось бы, 15 минут – это не так много. Однако следует учитывать количество моделей, которое необходимо подготовить за неделю с автоматизацией и без неё. Допустим, художник делает две модели в час. Так как он не будет тратить время на импорт, то к концу недели он сможет сделать около 80 моделей, которые будут готовы к использованию. Если же он будет тратить ещё 15 минут на импорт, то к концу недели будет выполнено 60 моделей. Разница в 20 моделей будет характерна для первой недели, со временем эта разница будет становиться всё больше. Таким образом, один довольно простой скрипт позволит существенно увеличить продуктивность и даст возможность укладываться в дедлайны.

Вышеприведённая реализация автоматизации показывает, что даже автоматизация небольшого пайплайна позволяет сэкономить огромное количество времени на длинной дистанции, ведь разработка игр занимает годы.

Такую автоматизацию пайплайна можно применить почти ко всей работе с внешними файлами, что приведёт к ускорению производства и стандартизации всего проекта.

В большинстве случаев проблемой переработок является неоптимизированный и переусложнённый ручной пайплайн, который приводит к потере огромного количества времени. Сотрудники боятся его менять, ведь смена пайплайна и ввод автоматизации могут привести к временному застою и к потере времени. Но как показывает практика, выигрыш от автоматизации очень существенен, и потеря времени от ввода нового пайплана компенсируется очень скоро. Поэтому при постоянных срывах сроков лучшим выходом будет не ввод переработок, а полное прекращение работы и анализ текущих рабочих процессов, нахождение новых оптимальных путей и реализация автоматизации в тех местах, где она сильно увеличит продуктивность как отдельного работника, так и всей команды в целом [10].

Библиографический список

1. Dietz J. 30 Games that emerged from development hell [Электронный ресурс]. URL: https://www.metacritic.com/feature/games-that-shed-vaporware-status (11.02.2021).
2. Semuels A. Every Game You Like Is Built on the Backs of Workers [Электронный ресурс]. URL: https://time.com/5603329/e3-video-game-creators-union/ (11.02.2021).
3. Weststar J., O'Meara V., Legault M-J. Developer Satisfaction Survey. 2017. 34 p. [Электронный ресурс]. URL: https://s3-us-east-2.amazonaws.com/igda-website/wp-content/uploads/2019/04/11143720/IGDA_DSS_2017_SummaryReport.pdf (15.02.2021).
4. Goldberg H. The Making of Red Dead Redemption 2 // Vulture [Электронный ресурс]. URL: https://www.vulture.com/2018/10/the-making-of-rockstar-games-red-dead-redemption-2.html (21.02.2021).
5. Schreier J. Video Games Are Destroying the People Who Make Them [Электронный ресурс]. URL: https://www.nytimes.com/2017/10/25/opinion/work-culture-video-games-crunch.html (21.02.2021).
6. Scheduled Overtime Effect on Construction Projects // A Construction Industry Cost Effectiveness Task Force Report. 1980. 18 p. [Электронный ресурс]. URL: https://kcuc.org/wp-content/uploads/2013/11/Scheduled-Overtime-Effect-on-Construction-Projects.pdf (14.03.2021).
7. Collewet M., Sauermann J. Working Hours and Productivity. Bonn: Institute of Labor Economics, 2017. 35 p. [Электронный ресурс]. URL: http://ftp.iza.org/dp10722.pdf (14.03.2021).
8. Cheong I.M. The Game Industry's Crunching Problem // Gameranx [Электронный ресурс]. URL: https://gameranx.com/features/id/18061/article/the-game-industry-s-crunching-problem-an-interview-with-stardock-s-derek-paxton/ (22.03.2021).
9. Unreal Engine 4 Python Documentation [Электронный ресурс]. URL: https://docs.unrealengine.com/en-US/PythonAPI/index.html (29.03.2021).
10. Coster S. Stress-Free Game Development // Informatech [Электронный ресурс]. URL: https://www.gdcvault.com/play/1026630/Stress-Free-Game-Development-Powering (30.03.2021).

Сведения об авторе / Information about the Author

Артемьев Владислав Александрович,
студент группы ЭВМм-19-1,
Институт информационных технологий и анализа данных,
Иркутский национальный исследовательский технический университет,
664074, г. Иркутск, ул. Лермонтова, 83, Российская Федерация,
e-mail: vladart4@gmail.com

Vladislav A. Artemiev,
Student,
Institute of Information Technology and Data Analysis,
Irkutsk National Research Technical University,
83 Lermontov Str., Irkutsk, 664074, Russian Federation,
e-mail: vladart4@gmail.com