УДК 004.94:794.02

ОНЛАЙН-ИНСТРУМЕНТ TULA ДЛЯ БАЛАНСИРОВКИ ВИДЕОИГР

В. Р. Рахманкулова[1] [0009-0006-8425-616X], В. В. Кугуракова[2] [0000-0002-1552-4910]

[1, 2]Казанский федеральный университет, *г. Казань, Россия*

[1]raxmankulova.v@mail.ru, [2]vlada.kugurakova@gmail.com

Аннотация

Разработан инструмент Tula для балансировки видеоигр. Его необходимость обоснована растущими требованиями к качеству и экономической эффективности в индустрии видеоигр, особенно в аспектах управления внутриигровой экономикой и логикой игрового мира. Проанализированы существующие инструменты и подходы к балансировке игр, выявлены их ограничения, на основе которых построен функционал нового инструмента, интегрирующего функции современных решений и предоставляющего расширенные возможности для анализа и тестирования игровых параметров, включая генерацию прототипов через описание классов и симуляцию в реальном времени. Описаны технологическая база и архитектура инструмента. Рассмотрены ключевые аспекты реализации: отзывчивость интерфейса, непрерывное обновление данных и безопасность. Проведенный сравнительный анализ с известным инструментом Machinations показал преимущества в корректности обработки данных, удобстве интерфейса и гибкости модификации прототипов.

Ключевые слова*: видеоигры, игровой процесс, игровые механики, игровой баланс, игровой дизайн, Machinations.*

ВВЕДЕНИЕ

С каждым днем требования к качеству и экономической эффективности создания видеоигр возрастают. Важной составляющей успеха в индустрии является как разработка общей логики игрового мира и игровых механик, влияющих на удержание внимания игроков, так и калибровка внутриигровой экономики и значений параметров игровых проектов, определяющие распределение ресурсов в игровом мире [1]. Поэтому особое значение приобретает балансировка числовых параметров всех игровых элементов. Цель представляемого проекта – упростить создание сбалансированных прототипов игровых

механик, а в рамках исследований по формализации игровых механик [2] – упростить создание сбалансированных прототипов видеоигр.

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ БАЛАНСА В ВИДЕОИГРАХ

Определение и анализ понятия

Понятие «игровой баланс» традиционно ассоциируется с равным распределением игровых ресурсов, обеспечивающим честную и мотивирующую игру. Существуют различные подходы к определению игрового баланса, поэтому имеет смысл привести некоторые их них, опираясь на последние исследования и практики в области разработки видеоигр.

В [3] на основе интервью с игровыми дизайнерами проанализирована концепция игрового баланса. Авторы подчеркивают важность достижения гармонии между интерактивностью, сюжетом и игровым прогрессом.

В [4] предложено выделение количественных параметров для оценки игр, значение которых зависит от контекста.

В [1] дан детальный анализ понятия игрового баланса с разграничением узкого математического понимания и более широкого дизайнерского представления.

В [5] представлены различные подходы практиков к балансировке, такие как акцентирование на принципе справедливости в многопользовательских играх, применение подхода к балансировке многопользовательских игр вокруг игрока, поиск и сохранение равновесия сил в игре.

Существует несколько широко известных научных работ, посвященных балансировке игр. Авторы этих работ используют или расширяют этот термин, часто принимая определения практиков [6], и оценивают динамические подходы к балансировке игры, утверждая, что они должны удовлетворять как минимум трем основным требованиям: адаптироваться к начальному уровню игрока, отслеживать эволюцию и регрессию его производительности, а также поддерживать игру «правдоподобной». Такая оценка показывает, что адаптивные подходы могут быть более эффективными и приводить к большей удовлетворенности пользователей, чем традиционная, неадаптивная, заранее заданная балансировка игры с использованием статических уровней сложности.

В многопользовательских играх неравенство в навыках может привести к разочарованию и нежелательным впечатлениям. Существуют подходы к балансировке для выравнивания игрового поля (например, предоставление игрокам с низкими показателями помощи в прицеливании), но неясно, как различные варианты дизайна влияют на индивидуальное восприятие игроков. В [7] предложено использовать пространство дизайна механик балансировки, а как один из итогов определена балансировка как неотъемлемая часть игры, не нарушающая основной геймплей.

По нашему мнению, не отрицая предлагаемые подходы теоретиков и практиков геймдизайна, необходимо подчеркнуть, что определение игрового баланса непосредственно тесно связано с процессом проектирования механик игры. И все же в узком смысле понимания балансировки игрового баланса можно рассматривать как методику настройки и анализа числовых параметров элементов прототипа игровой механики. Основная его цель – гарантировать, что идеи геймдизайна будут не только реализованы, но и функционально достижимы.

Сравнительный анализ существующих инструментов

Анализ часто используемых инструментов для разработки видеоигр позволяет выделить как их преимущества, так и ограничения.

Бесплатные инструменты для обработки данных в виде таблиц (например, популярный инструмент Google Sheets[1]) предоставляют базовые возможности, включая разнообразную обработку данных. Однако, несмотря на широкие аналитические возможности, эти инструменты обладают ограниченным функционалом для разработки прототипов игровых механик из-за их шаблонной и строго математической природы. С развитием и усложнением игровых прототипов становится очевидной потребность в более интуитивно понятных и визуально ориентированных инструментах.

[1] Google Sheets – spreadsheet application included as part of the free, web-based Google Docs Editors suite offered by Google. https://www.google.com/sheets/

В отличие от табличных приложений, завоевавшая бесспорную популярность в своем сегменте бесконечная интерактивная онлайн-доска Miro[2] имеет ограничения в анализе данных и разработке сложных моделей игровых механик. Хотя инструмент Miro очень эффективен для визуального прототипирования в режиме совместной работы, он менее подходит для обработки больших схем или выполнения сложных расчетов.

Браузерный инструмент Machinations[3], разработанный специально для проектирования и балансировки игровых прототипов, позволяет моделировать игровые процессы, проводить эксперименты и собирать количественные данные из симуляций игровых механик и систем, отображая результаты в виде диаграмм. Сравнение методов документирования требований к поведению NPC (Non-Player Character — неигровой персонаж в видеоиграх) показало, что использование динамических диаграмм в Machinations значительно ускоряет процесс работы над проектом, уменьшая необходимость в заполнении объемных текстовых документов [8].

Однако, несмотря на неоспоримые преимущества, Machinations имеет ряд ограничений, таких как:

- проблемы с производительностью, визуализацией и масштабированием сложных прототипов;
- сложность в понимании взаимосвязей между сущностями из-за их комплексной структуры;
- вероятность ошибок при запуске симуляций из-за отсутствия проверки валидности числовых значений в полях;
- невозможность внесения изменений во время симуляции;
- ограниченное количество шагов, доступных для проведения симуляции, сужающее функционал;
- сложный и перегруженный интерфейс.

[2] Miro – visual workspace where teams manage projects, design products. https://miro.com/
[3] Machinations – game design tool for balancing. https://machinations.io/

КОНЦЕПЦИЯ И АРХИТЕКТУРА НОВОГО ИНСТРУМЕНТА

Принципы проектирования

Из анализа существующих инструментов стало очевидным, что для разработки будущего инструмента критично важно обеспечить простую и понятную визуализацию данных для прототипирования, а также интегрировать функционал для анализа и сравнения метрик. Предлагаемый нами инструмент объединяет лучшие функции современных решений, предоставляя интуитивно понятный интерфейс и расширенные возможности для тестирования и анализа игровых механик. Дополнительные возможности итогового приложения включают:

- гибкая настройка весов, позволяющая регулировать различные игровые параметры, такие как скорость, сложность и уровень ресурсов, улучшая тем самым гибкость проектирования;
- визуализация игровых элементов в виде узлов и связей между ними, что облегчает восприятие и анализ структуры игры;
- тестирование в реальном времени – функция, позволяющая наблюдать за изменениями в динамике игры напрямую во время тестирования прототипа;
- аналитические инструменты, которые позволяют изучать эффективность игровых стратегий и оптимизировать распределение ресурсов для улучшения баланса и интерактивности игры;
- функциональность для генерации прототипов видеоигр на основе описания классов игровых механик в онлайн-редакторе кода;
- поддержка экспорта данных, например, в формат JSON.

Предлагаемый инструмент опирается на формальный подход к пространственно-временному моделированию игровых систем [2], что обеспечивает математически обоснованную балансировку параметров.

Сценарии работы

На основе спроектированного концепта инструмента был выстроен следующий сценарий работы пользователя.

- **Авторизация:** Пользователь вводит данные для аутентификации через электронную почту или аккаунт на GitHub[4].
- **Выбор доски:** После входа в систему пользователь переходит на личную страницу, где выбирает доску для начала работы.
- **Выбор режима:** Пользователь выбирает режим построения прототипа – ручной или генерация путем описания классов в онлайн редакторе.
- **Построение прототипа:** Пользователь создает прототип механики игры или редактирует сгенерированный прототип, перетаскивая узлы и устанавливая связи между ними на интерактивной доске, при этом вводя числовые характеристики для симуляции.
- **Настройка параметров аналитики**: Пользователь выбирает количество итераций, время в секундах и количество игровых сессий для запуска симуляций путем заполнения соответствующих полей на интерактивной панели.
- **Выполнение симуляции**: Пользователь запускает симуляцию прототипа и выбирает нужный узел, «щелкая» по нему для анализа значений узла в результате пройденной игровой сессии.
- **Анализ результатов**: Пользователь анализирует полученные значения и подбирает необходимую числовую характеристику для дальнейшей работы.
- **Выгрузка данных**: По завершении анализа пользователь может выгрузить код в формате JSON или схему в расширении PNG для дальнейшей разработки игры.

В это время инструмент работает по следующему сценарию.

1. **Аутентификация**: Инструмент проверяет предоставленные учетные данные, отправляя запрос в систему Clerk[5], и в случае успешной аутентификации предоставляет доступ к функционалу инструмента.

[4] GitHub – облачная платформа для хостинга IT-проектов и совместной разработки.
[5] Clerk – authentication and user management system. https://clerk.com

2. **Личные доски**: При входе инструмент предлагает выбрать доску из существующих или же создать новую, а также добавить доску в избранное. Система отправляет запрос в систему Clerk, сопоставляя токен пользователя с доступными ему досками.
3. **Коллаборация**: При выборе доски инструмент предлагает возможность пригласить других участников для совместной работы над выбранной доской путем интеграции с библиотекой Liveblocks[6] и системой Clerk для авторизованных аккаунтов.
4. **Выбор режима**: Инструмент реагирует на запрос пользователя о выборе режима построения прототипа, предоставляя возможность выбора между ручным режимом и генерацией на основе описания классов в онлайн редакторе.
5. **Создание прототипа**: Инструмент предоставляет интерфейсную панель на интерактивной доске для создания узлов.
6. **Настройка параметров аналитики**: Инструмент сохраняет введенные значения в менеджере состояний[7] и автоматически запускает симуляцию прототипа с заданными параметрами через установленный интервал.
7. **Выполнение симуляции**: Система временно сохраняет данные после каждой симуляции игрового прототипа в менеджер состояний. По завершении каждой игровой сессии она отправляет POST-запрос на сервер, предоставляя информацию о значениях узлов на каждом шаге итерации для последующего анализа, вся информация сохраняется в базе данных.
8. **Анализ значений**: После выбора узла путем клика по нему инструмент отправляет GET-запрос на сервер с идентификатором узла, запрашивая

[6] Liveblocks – real-time collaboration infrastructure for building performant collaborative experiences. https://liveblocks.io/

[7] Менеджер состояний (statemanager) – это инструмент, используемый в программировании для управления состоянием приложения. В контексте веб-разработки и разработки приложений состояние относится к любым данным или информации, которые могут изменяться во время работы приложения, например это данные пользователя, настройки интерфейса, информация о текущей странице и т.д.

информацию. Система сопоставляет значения запроса с информацией в базе данных и затем отправляет данные клиенту, который, в свою очередь, отображает полученные данные в виде линейной диаграммы.

9. **Предоставление данных**: Инструмент предоставляет пользователю возможность анализировать полученные результаты и выгружать для дальнейшей работы код в JSON или схему в PNG. Выбор формата JSON обусловлен несколькими причинами, в частности тем, что он является широко используемым и понятным форматом данных, а его использование в качестве формата экспорта механики игры позволяет значительно сократить время, затрачиваемое на документирование игровой системы.

Описанный сценарий работы созданного инструмента представляет собой четко структурированный процесс, который обеспечивает удобство и гибкость при создании и анализе прототипов.

Архитектурные решения

Переходя к описанию технологической базы инструмента, остановимся на выборе программных платформ и методов обработки данных. В процессе проектирования архитектуры интерактивного приложения возник ряд типичных проблем, таких как:

- отзывчивость интерфейса и быстрая загрузка данных – важно, чтобы переходы между экранами и загрузка данных происходили быстро, чтобы обеспечить плавное взаимодействие пользователя с приложением и избежать задержек;
- непрерывное обновление данных – данные в приложении должны обновляться быстро и незаметно для пользователя, чтобы не нарушать его работу с приложением.

Для бесшовного решения этих проблем предложена следующая архитектура (рис. 1).

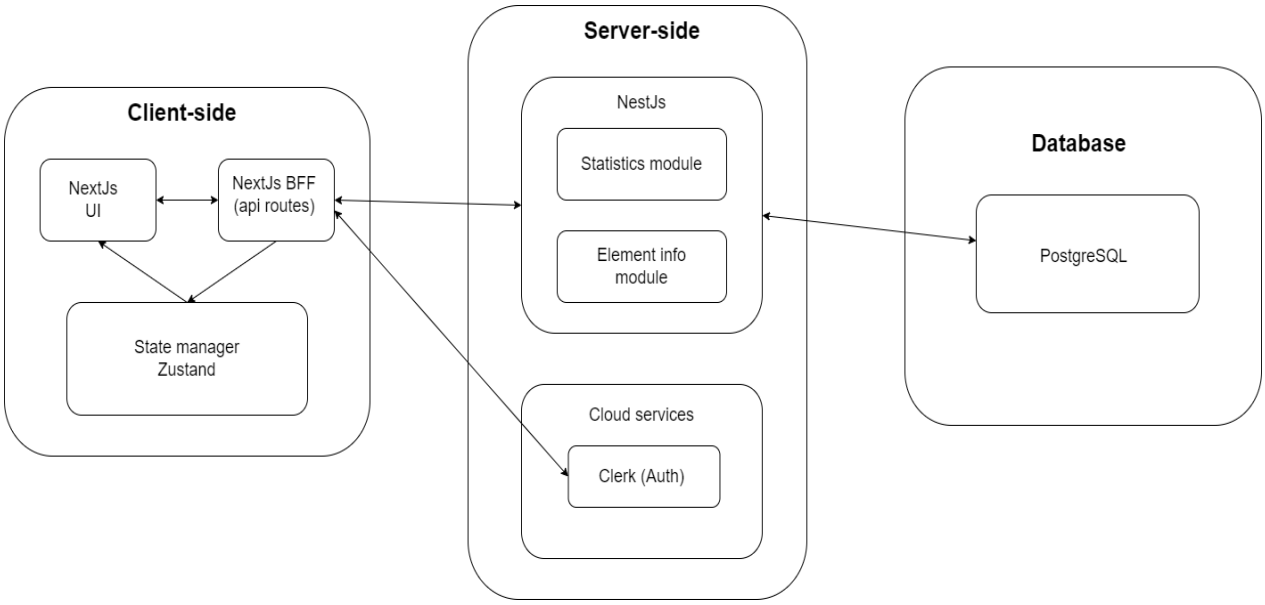

Рис. 1. Архитектура онлайн-инструмента

Клиентская часть инструмента разработана с использованием фреймворка Next.js[8], который обеспечивает удобную интеграцию с разнообразными библиотеками для создания пользовательских интерфейсов и применяет подход BFF[9], предоставляя каждому интерфейсу возможность взаимодействовать с серверным компонентом через API для обеспечения необходимой функциональности. Next.js позволяет реализовать предварительную загрузку страниц и асинхронную загрузку данных, что способствует быстрым переходам между экранами и минимизации времени ожидания.

Для глобального управления состоянием использована библиотека Zustand[10], предназначенная для хранения и обновления временных данных в процессе работы инструмента. Благодаря минималистичному синтаксису Zustand создание модульных хранилищ данных для различных компонентов происходит без создания громоздкой архитектуры, позволяя компонентам обновляться автоматически без явного вызова методов перерисовки.

[8] Next.js – React Framework for the web applications documentation. https://nextjs.org/

[9] BFF (сокр. от англ. Backend For Frontend) – архитектурный подход, позволяющий разделить фронтенд и бэкенд.

[10] Zustand – state manager for the web applications documentation. https://docs.pmnd.rs/zustand/

На серверной стороне для повышения безопасности внедрен инструмент Clerk, механизм которого позволяет выполнять аутентификацию и проверять доступ пользователя к определенным страницам до полной загрузки контента.

В качестве фреймворка для выполнения математических вычислений, расчета статистики и обработки информации об игровых элементах выбран Nest.js.

Для обработки и хранения данных задействована база данных PostgreSQL[11], к которой происходит обращение через Nest.js с использованием ORM Prisma[12]. PostgreSQL обеспечивает надежное хранение данных о состояниях игровых элементов, получаемых в ходе симуляций, что позволяет эффективно анализировать и использовать эту информацию для статистических расчетов.

РЕАЛИЗАЦИЯ И ФУНКЦИОНАЛЬНОСТЬ ИНСТРУМЕНТА

В качестве имени разработанного инструмента выбрано рабочее название, отражающее главную цель, – балансирование игровых параметров. На санскрите *тула* ((तुल [13], Tula) означает «весы» или «баланс», и это слово можно транслитерировать на латиницу как Tula.

Особенности реализации

После входа в систему пользователю представляется главная страница инструмента (рис. 2).

[11] PostgreSQL – open source database documentation. https://www.postgresql.org/

[12] ORM (сокр. от англ. Object-Relational Mapping) Prisma – объектно-реляционное отображение, позволяющее работать как с реляционными (PostgreSQL, MySQL, SQL Server, SQLite), так и нереляционной (MongoDB) базами данных с помощью JavaScript или TypeScript без использования SQL.

[13] Деванагари — слоговое письмо, используемое для записи санскрита, хинди и других языков Индии.

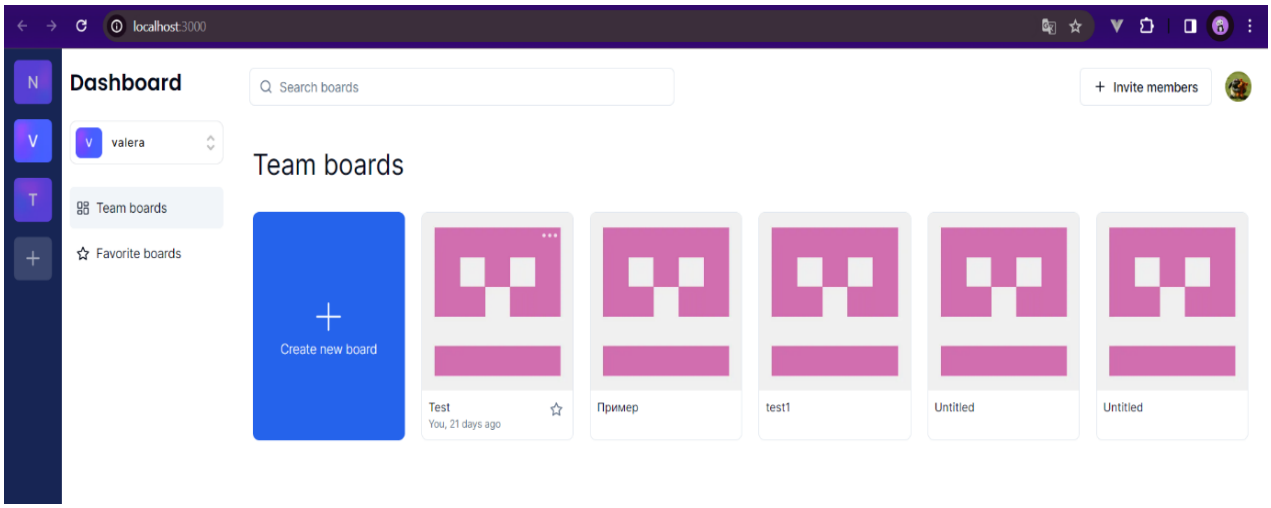

Рис. 2. Главная страница инструмента Tula

Боковая панель отображает список команд, что облегчает разделение задач между разными проектами. На этой панели можно создавать отдельные доски для каждой команды, приглашать участников для совместной работы и эффективно управлять коллаборацией. Доски предназначены для последующей работы и управления проектами.

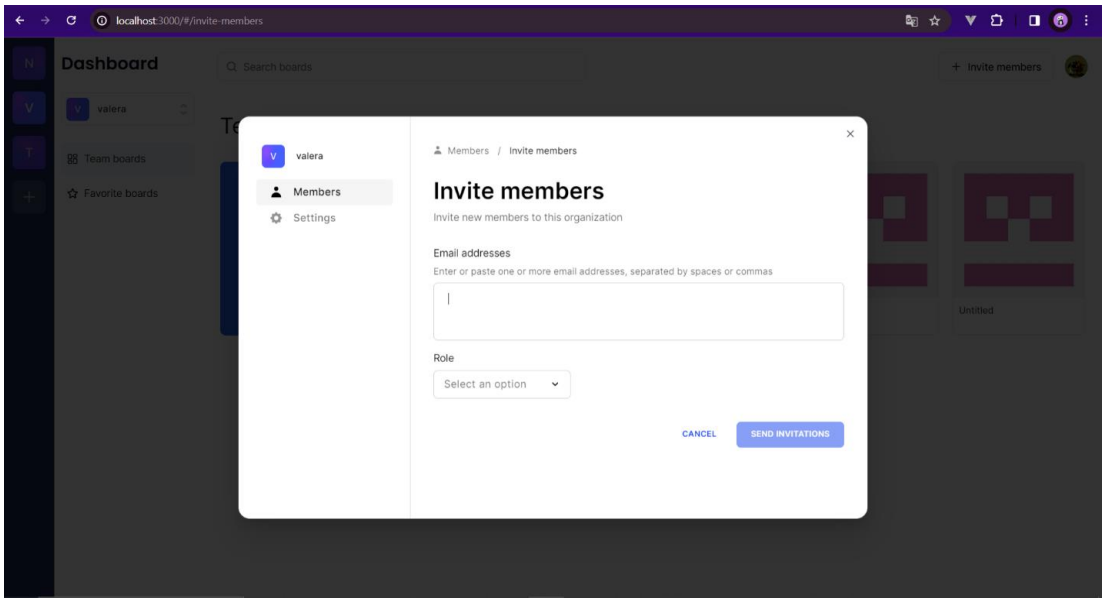

Рис. 3. Окно для отправки приглашений

Создав доску, пользователь имеет возможность отправить приглашения другим пользователям для совместной работы (рис. 3). Инструмент поддерживает реализацию совместной работы в режиме реального времени через

WebSocket[14]. Эта функция синхронизирована с Liveblocks для гарантии стабильности.

Архитектурно система автоматически управляет переходом в оффлайн-режим, восстановлением связи после ее потери и другими подобными сценариями, обеспечивая непрерывное функционирование.

Графовая система узлов

Игровая механика и симуляция изображаются с помощью диаграмм. Инструмент предоставляет пользователю визуальный язык для описания взаимодействия игровых элементов. Этот язык, базирующийся на наборе узлов, был разработан с учетом опыта использования схем в инструменте Machinations. Пользовательские отзывы (например, в [9]) подчеркивали сложность восприятия схем в Machinations, что стало толчком для внесения улучшений и будет описано ниже при проведении сравнения.

На левой панели инструмента (рис. 4) представлены иконки и описания компонентов, используемых в игровых схемах.

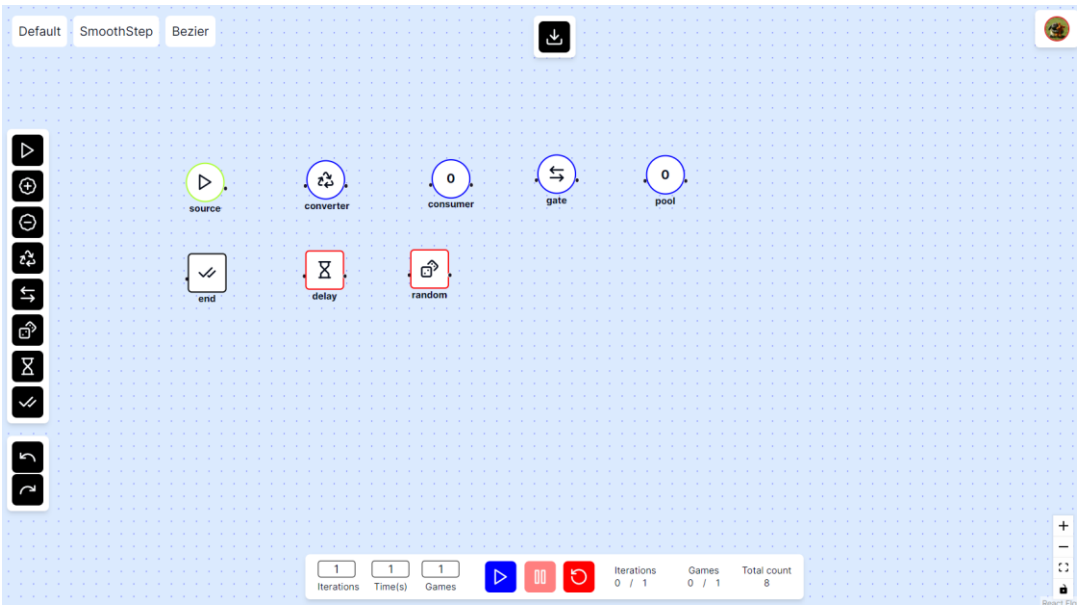

Рис. 4. Интерфейс доски

[14] WebSocket – технология, которая позволяет клиенту установить двухстороннюю («дуплексную») связь с сервером.

Каждый узел выполняет определенную функцию в механике игры:

- Source – генерирует ресурсы;
- Pool – служит для хранения ресурсов;
- Consumer – потребляет ресурсы;
- Converter – преобразует один вид ресурса в другой;
- Gate – активирует процессы при выполнении заданных условий;
- Random – инициирует действия на основе случайных событий;
- Delay – обеспечивает временную задержку в процессах;
- End – обозначает конечную точку игры.

Представленный язык спецификации обладает интуитивной простотой, что облегчает его освоение будущими пользователями. Это делает его доступным для широкого круга участников разработки разной направленности.

Для улучшения визуализации механики игры предоставлены три вида соединений, позволяющие связывать компоненты путем перетаскивания ручек от одного элемента к другому (рис. 5–7).

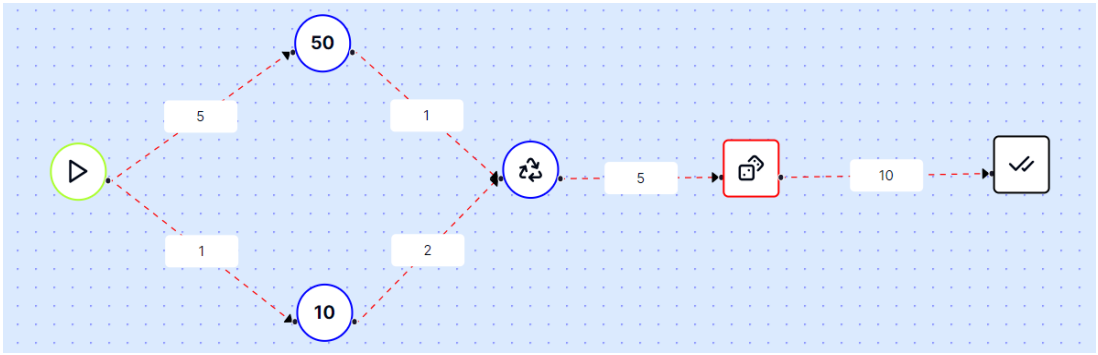

Рис. 5. Соединение по кратчайшему пути

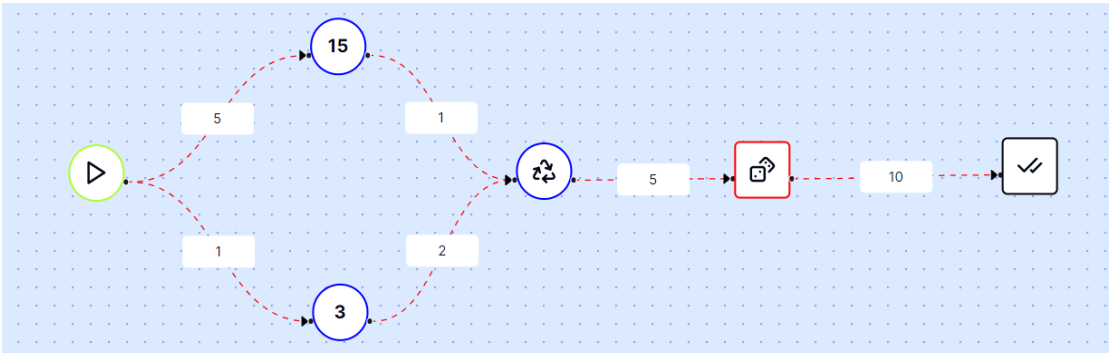

Рис. 6. Соединение кривой Безье

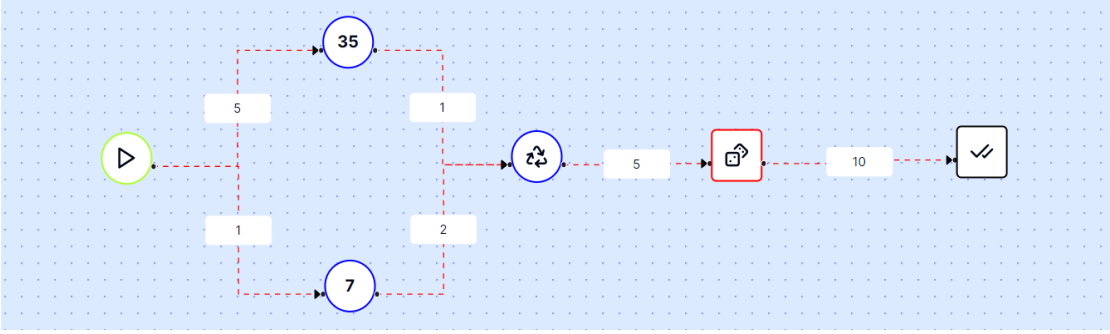

Рис. 7. Соединение прямыми линиями

Разработанный функционал позволяет гибко настраивать тестовые условия, включая количество шагов и продолжительность выполнения симуляции (рис. 8).

Рис. 8. Панель для запуска симуляций

Далее опишем панель подробнее.

- Первое поле служит для установки количества итераций, которые должна выполнить система в течение одной игровой сессии.
- Второе поле задает продолжительность одной итерации в секундах.
- Третье поле определяет количество игровых сессий, которые предстоит симулировать. Каждая сессия включает в себя ранее заданное количество итераций, что позволяет анализировать разнообразные распределения ресурсов в ходе игры.
- Кнопки управления на панели позволяют начать визуализацию симуляции, приостановить и возобновить процесс на определенном этапе. Последняя кнопка предназначена для сброса всех данных и начала симуляции с первой итерации первой сессии.
- Боковая часть панели отображает счетчик выполненных итераций и игровых сессий, а также показывает общее количество элементов, используемых в схеме, включая соединения и компоненты.

Алгоритм случайного распределения ресурсов

В разработке игровых систем ключевую роль играет учет случайных событий, которые вносят элемент непредсказуемости в игровой процесс.

Для более точного моделирования реальных условий игры был разработан компонент, который позволяет случайно распределять ресурсы (рис. 9). Этот компонент дает возможность задавать количество ресурсных единиц для распределения на входе. На выходе он связывается с другими элементами, которые принимают ресурсы с разной вероятностью. Соединения, выходящие из этого компонента, показывают пропорции распределения ресурсов. Например, при распределении 10 единиц ресурса 1/10 единицы перенаправляется по первому соединению, 2/10 – по второму и третьему, а 5/10 – по четвертому.

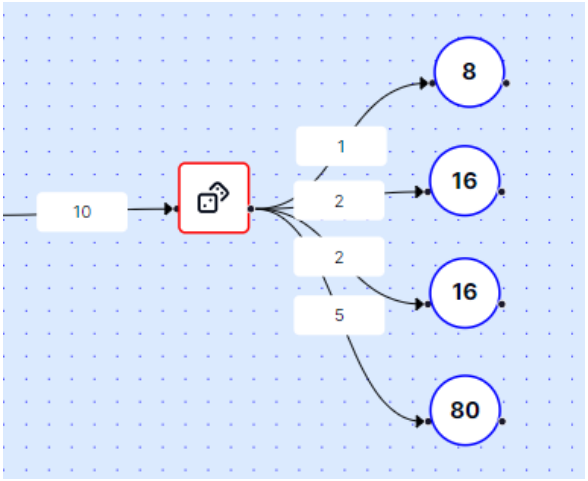

Рис. 9. Компонент для случайного распределения

Алгоритм случайного распределения ресурсов состоит из следующих этапов:

1) вычисление вероятностей;
2) распределение значений по вероятности;
3) распределение оставшихся ресурсов;
4) фиксация результата распределения.

На этом этапе вычисляется вероятность отправки ресурса по каждой из ветвей. Для этого сначала суммируются значения всех ребер, чтобы получить общее значение, которое доступно для распределения. Затем каждое значение ветви делится на общее значение, чтобы определить вероятность для каждой ветви

$$p_i = e_i / \sum_{i=1}^{n} e_i,$$

где i – натуральное число, p_i – вероятность отправки ресурса по i-й ветви, а e_i – значение i-го ребра.

Далее происходит распределение ресурсов с учетом вычисленных вероятностей:

1) генерируется случайное число от 0 до оставшегося количества ресурсов;
2) полученное число умножается на вероятность определения количества ресурсов, которые будут отправлены по этой ветви; результат округляется в меньшую сторону с помощью функции Math.floor;
3) определенное количество ресурсов вычитается из общего доступного количества и отправляется по текущей ветви;
4) процесс повторяется для каждой ветви.

Если после основного распределения остались нераспределенные ресурсы, они равномерно распределяются между всеми ветвями. Дополнительно используются случайные числа для обеспечения разнообразного и случайного распределения оставшихся ресурсов.

После завершения всех этапов распределения возвращается массив, где каждый элемент представляет количество ресурсов, отправленных по каждой ветви.

Аналитические возможности

Для визуализации потенциальных исходов игровой сессии использована линейная диаграмма. Инструмент автоматически рассчитывает среднее, медианное, минимальное и максимальное значения для каждого узла на каждом этапе итерации, что позволяет проводить всесторонний анализ баланса:

- среднее значение: позволяет оценить общую тенденцию изменений во время симуляции, отражая основные тенденции данных;
- медианное значение: помогает выявить аномалии и сравнить их со средним значением, что способствует обнаружению выбросов;
- минимальное и максимальное значения: использованы для определения диапазона возможных изменений значений, что критически важно для понимания крайних границ изменений параметров.

Выполнение итераций и симуляций дает возможность вносить изменения в числовые характеристики системы, проводить эксперименты и детально анализировать получаемые результаты, что обеспечивает основу для информированного принятия решений по настройке баланса игровых параметров.

СРАВНЕНИЕ С MACHINATIONS

Практическое применение

Демонстрацию работы инструмента на примере создания прототипа игровой механики, включая настройку параметров и анализ результатов, можно сопроводить сравнением функциональности с аналогичными решениями в Machinations.

В качестве примера рассмотрим механику производства, в которой игрок получает деревья и камень для строительства домов.

1) Должен быть создан источник генерации ресурсов, который описывает добычу материалов игроком.
2) Должны быть добавлены в систему два узла для хранения ресурсов (Камни и Дерево).
3) Конвертер «Построить Дом» будет преобразовывать ресурс «Дерево» в «Дома».

4) Нужно установить связи между узлами и указать количество единиц ресурса, участвующих в игровой механике.

Дадим общее описание механики: игрок за одну игровую итерацию добывает 1 единицу Wood (Дерево) и 2 единицы Stone (Камень). Когда количество Дерева достигает 5 единиц, функция Build House (Построить Дом) извлекает 5 ресурсов из пула Дерево и преобразует их в один ресурс Дома. Этот новый ресурс хранится в пуле построенных Домов.

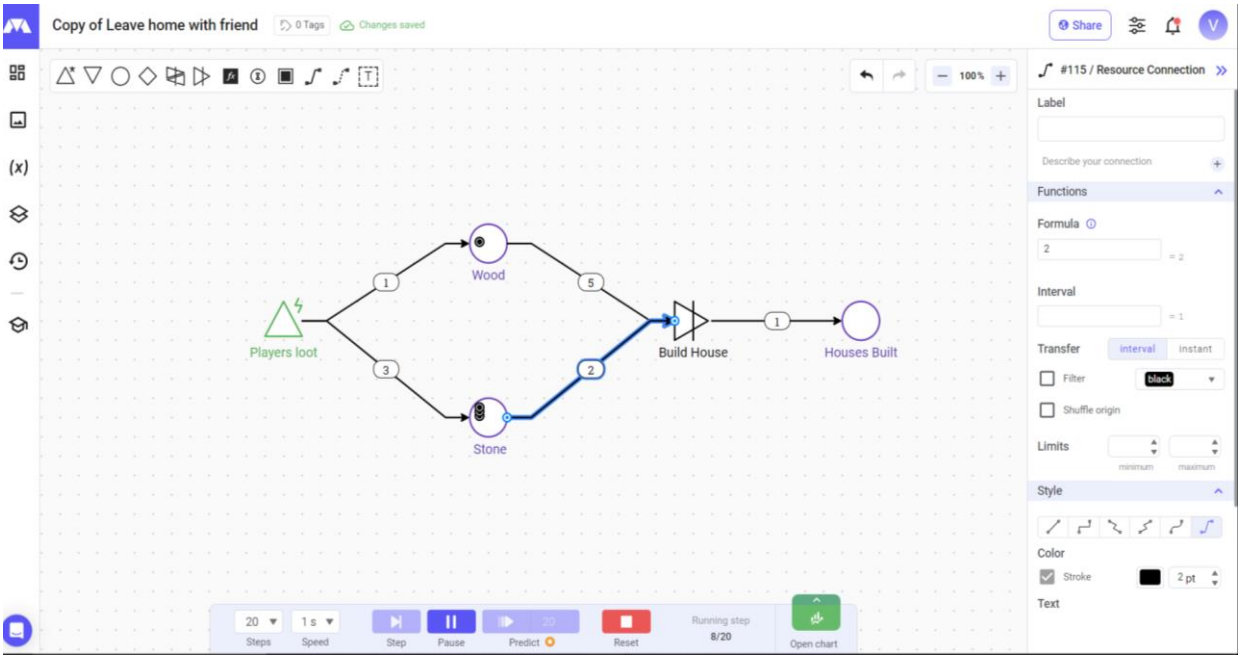

Рис. 10. Прототип игровой механики в Machinations

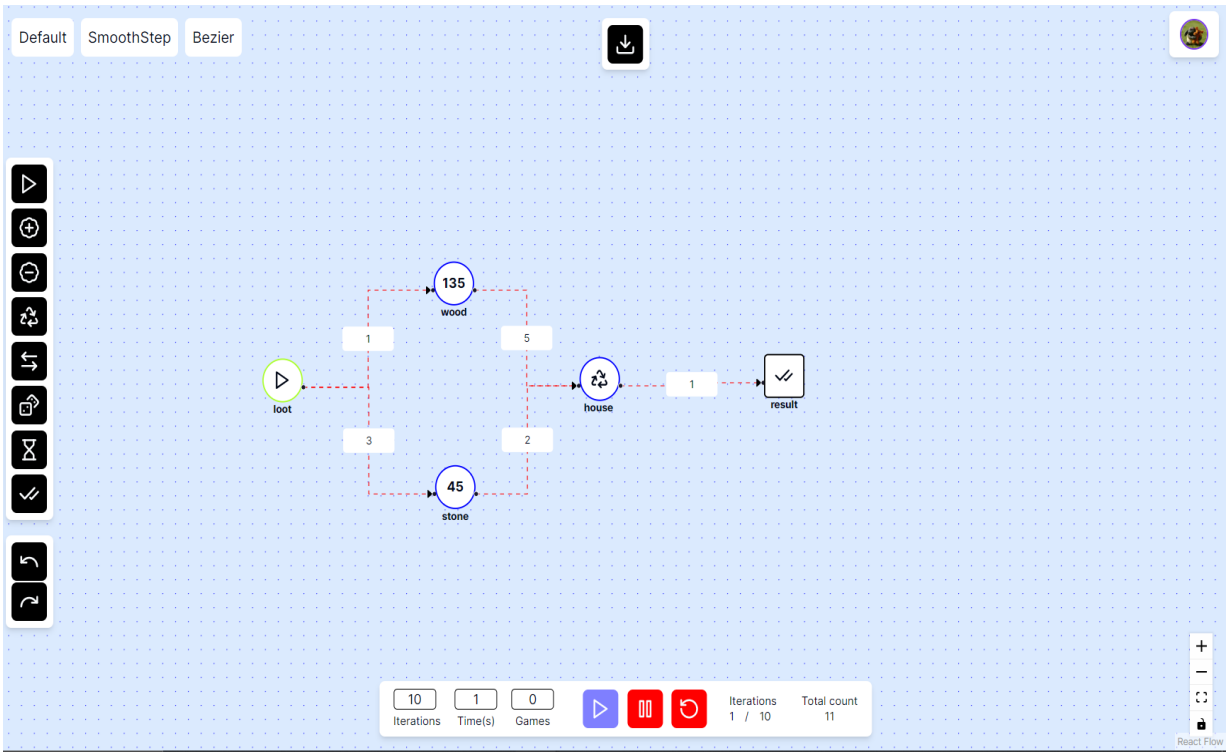

Рис. 11. Прототип игровой механики в Tula

Было принято решение убрать описательную панель, которая находилась справа от интерфейса (см. рис. 10 и 11), после того как были обнаружены проблемы с ее громоздкостью и неэффективным использованием.

В ходе улучшений системы внедрены дополнительные функциональные возможности, в том числе отмена последнего действия и возврат к предыдущему состоянию. Эти возможности осуществляются благодаря механизму хранения истории операций, реализованному с помощью стейтменеджера, что обеспечивает необходимые инструменты для создания прототипов игровых механик.

Валидация полей

Отсутствие валидации полей для ввода значений является существенным недостатком инструмента Machinations, обнаруженным в ходе проектирования. Система позволяет вводить текстовые значения (рис. 12) без какой-либо проверки или уведомления об ошибке, что приводит к запуску игровой симуляции с неверными данными и к некорректному поведению системы.

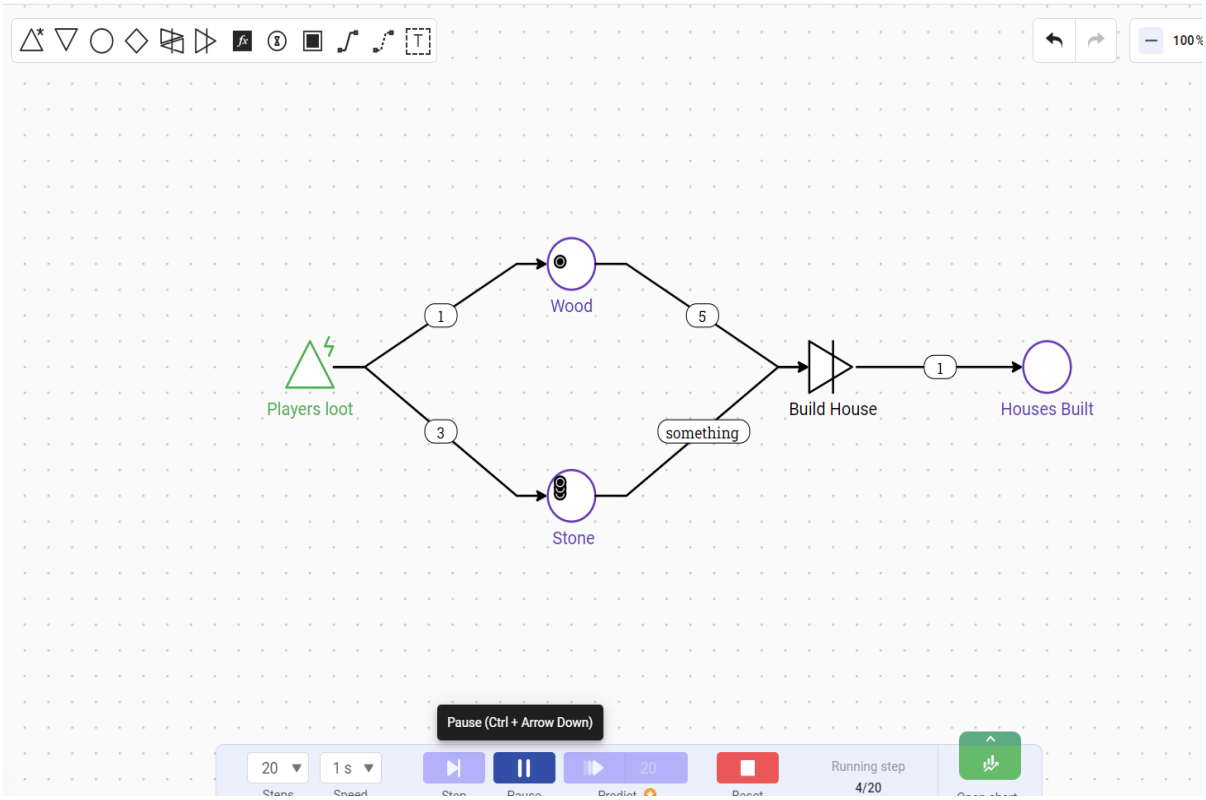

Рис. 12. Невалидные значения (в Machinations)

В инструмент Tula был внедрен механизм проверки валидации строковых значений. Если в поле ребра указано нечисловое значение, система отображает ошибку “Must be a numeric!” и блокирует возможность запуска симуляции (рис. 13). Кнопка запуска остается неактивной до тех пор, пока некорректное значение не будет исправлено.

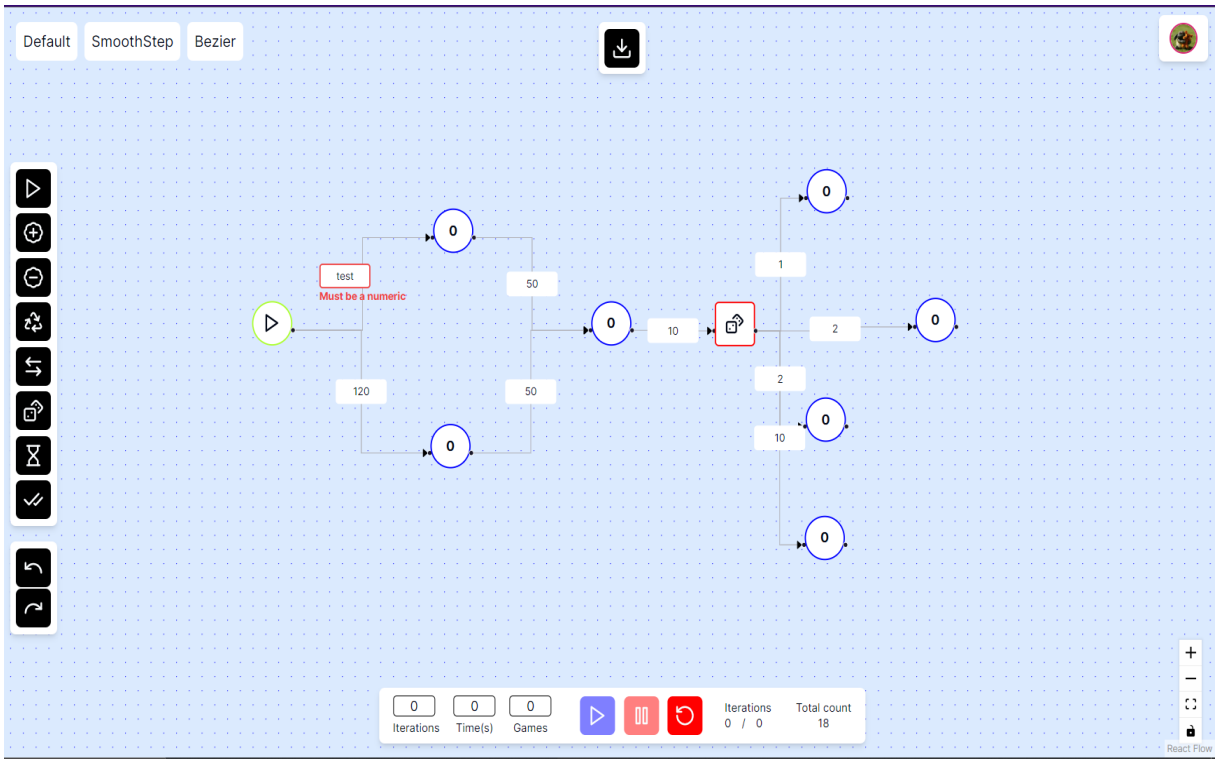

Рис. 13. Невалидные значения (в Tula)

Удаление узлов

Частые изменения в прототипах – это нормальное явление. Поэтому важно иметь возможность быстро удалять узлы и связи между ними.

В инструменте Machinations при удалении узлов (см. рис. 14) связи остаются на месте.

Если узел удаляется из середины цепочки, связь между смежными узлами не обновляется автоматически, что требует ручного удаления и создания новых связей, что отнимает много времени.

В разработанном инструменте Tula эта проблема была решена: после удаления узла связи автоматически пересчитываются и обновляются (рис. 15).

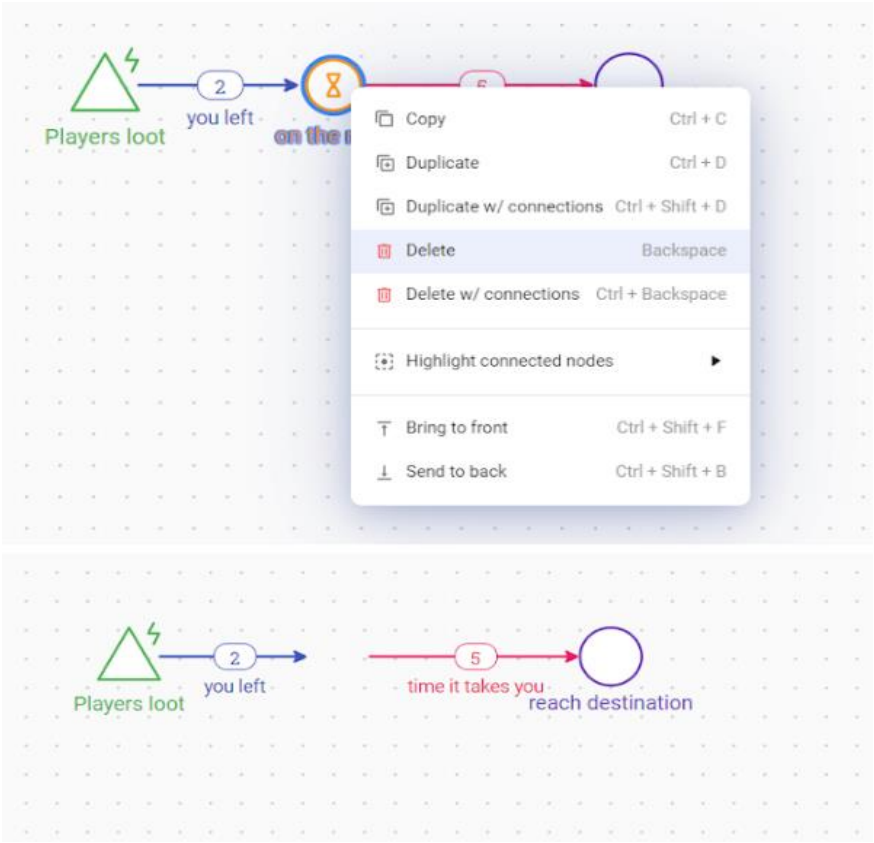

Рис. 14. Удаление узлов в Machinations

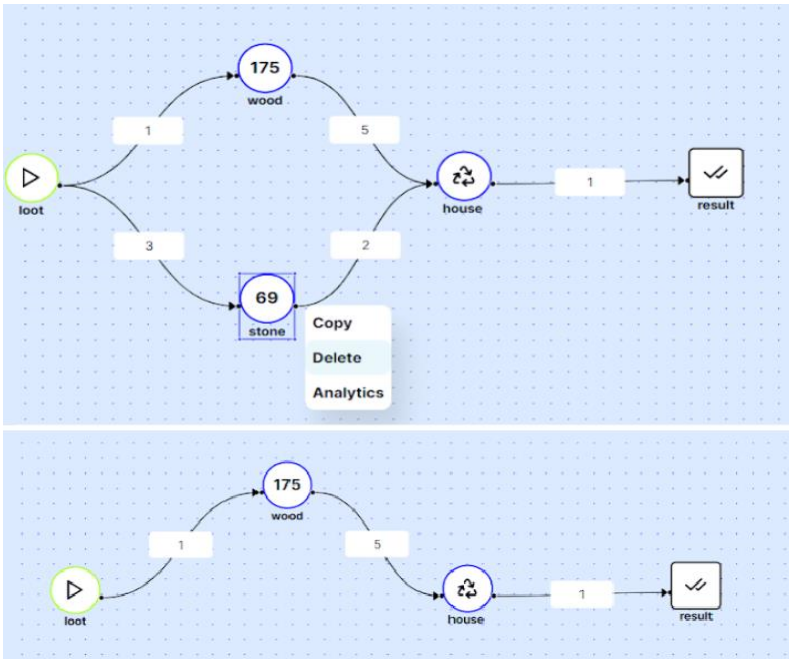

Рис. 15. Удаление узлов в Tula

Анализ значений

При анализе функциональности инструмента Machinations выяснилось, что пользователи часто сталкиваются со сложностью и избыточным объемом информации на аналитических диаграммах (рис. 16), что усложняет их понимание. С учетом отмеченной проблемы процесс вывода результатов аналитических диаграмм был упрощен (рис. 17), что делает их более доступными и понятными для пользователей.

Рис. 16. Результаты анализа значений игрового узла в Machinations

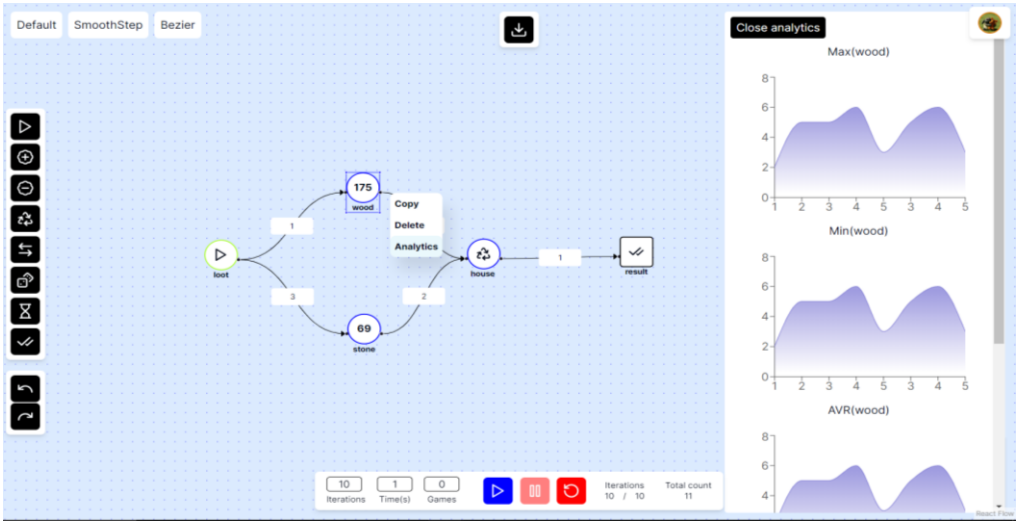

Рис. 17. Результаты анализа значений игрового узла в Tula

Технология также была улучшена путем добавления возможности отображения базовой диаграммы одним кликом на соответствующий узел после этапа игровой сессии для проведения анализа. Это позволяет пользователям мгновенно просматривать диаграмму, которая ясно указывает анализируемый узел и отображает выходные значения в форме понятных диаграмм.

Дополнительный функционал

В рамках созданной системы планируется разработка функционала, который облегчит генерацию первичной структуры игровой механики, на основе описания классов как на популярных языках программирования (например, JavaScript), так и при помощи формального описания (обычным текстом или, например, в JSON). Благодаря такому механизму будет предоставлен удобный кодовый редактор для создания исходной схемы игровой механики с использованием уже существующих описаний элементов.

Эта инициатива находит подтверждение в [3], где подробно рассмотрен потенциал прототипирования игр с применением семантического анализа и машинного обучения на базе текстовой документации. Информация, содержащаяся в документации, может быть эффективно использована для автоматической генерации первичного прототипа.

ЗАКЛЮЧЕНИЕ

Представлен подробный анализ и разработан инструмент для балансировки видеоигр. Продемонстрированы теоретические основы, практическое применение и сравнительный анализ, подчеркивающий значимость инструмента для индустрии видеоигр. Учитывая представленные перспективы развития реализованного инструмента TULA, мы видим его потенциал для дальнейшего улучшения и расширения функциональности. Развитие инструмента сыграет важную роль в индустрии разработки видеоигр, что обеспечивает разработчиков более эффективными средствами для создания сбалансированных видеоигр. Разработанный инструмент TULA является частью более широкого исследования по формализации игровых механик и их верификации. В

перспективе планируется его интеграция с методами автоматической генерации прототипов на основе формальных спецификаций [2], что позволит создать комплексную среду разработки и балансировки игр.

Благодарности

Работа выполнена за счет средств Программы стратегического академического лидерства Казанского (Приволжского) федерального университета («ПРИОРИТЕТ-2030»).

СПИСОК ЛИТЕРАТУРЫ

1. *Сахибгареева Г.Ф., Кугуракова В.В., Большаков Э.С.* Инструменты балансирования игр // Электронные библиотеки. 2023. Т. 26, № 2. С. 225–251.

2. *Кугуракова В.В.* Формальный подход к пространственно-временному моделированию игровых систем // Ученые записки Казанского университета. Серия Физико-математические науки. 2024. Т. 166. № 4. С. 532–554.

3. *Rouse R.* Game Design: Theory and Practice: Theory and Practice. Jones & Bartlett Learning, 2004.

4. Game Balance Concepts [Электронный ресурс]. URL: https://gamebalanceconcepts.wordpress.com (17.09.25)

5. *Becker A., Görlich D.* What is game balancing? An examination of concepts // ParadigmPlus. 2020. Vol. 1, No. 1. P. 22–41.

6. *Andrade G. et al.* Dynamic game balancing: An evaluation of user satisfaction // Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2006. Vol. 2, No. 1. P. С. 3–8.

7. *Gonçalves D. et al.* The Trick is to Stay Behind?: Defining and Exploring the Design Space of Player Balancing Mechanics // Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. 2024. P. 1–16.

8. *Сахибгареева Г.Ф., Кугуракова В.В., Большаков Э.С.* Генерация и балансирование игровых механик видеоигр // Научный сервис в сети Интернет: труды XXIV Всероссийской научной конференции. 2022. С. 455–485.

9. *Галимзянов Г.Р*. Разработка инструмента автоматической корректировки внутриигровых параметров: выпускная квалификационная работа 09.03.04. Казань, 2021. 35 с.

TULA ONLINE TOOL FOR BALANCING VIDEO GAMES

V. R. Rakhmankulova[1] [0009-0006-8425-616X], V. V. Kugurakova[2] [0000-0002-1552-4910]

[1, 2]Institute of Information Technologies and Intelligent Systems, Kazan Federal University, 35 Kremlyovskaya st., Kazan, 420008

[1]raxmankulova.v@mail.ru, [2]vlada.kugurakova@gmail.com

Abstract

This paper presents the development of Tula, a tool for video game balancing. The necessity for such a tool is substantiated by the growing requirements for quality and cost-effectiveness in the video game industry, particularly in managing in-game economy and game world logic. The study analyzes existing tools and approaches to game balancing, identifying their limitations, which informed the design of the new tool's functionality. The presented tool integrates features of contemporary solutions while providing enhanced capabilities for game parameter analysis and testing, including prototype generation via class descriptions and real-time simulation. The technological foundation and architecture of the tool are described in detail. Key implementation aspects are discussed: interface responsiveness, continuous data synchronization, and security. Comparative analysis with Machinations revealed advantages in data processing correctness, interface convenience, and prototype modification flexibility.

Keywords*: video games, gameplay, game mechanics, game balance, game design, Machinations.*

Acknowledgment: This paper has been supported by the Kazan Federal University Strategic Academic Leadership Program ("PRIORITY-2030").

REFERENCES

1. *Sahibgareeva G.F., Kugurakova V.V., Bolshakov E.S.* Video Game's Mechanics Generation and Balancing // In CEUR Workshop Proceeding, 2022. P. 455–485.

2. *Kugurakova V.V.* A Formal Approach to Spatio-Temporal Modeling of Game Systems // Uchenye Zapiski Kazanskogo Universiteta. Seriya Fiziko-Matematicheskie Nauki. 2024. Vol. 166, Is. 4. P. 532–554 (In Russ.)

3. *Rouse R.* Game Design: Theory and Practice: Theory and Practice. Jones & Bartlett Learning, 2004.

4. Game Balance Concepts.
URL: https://gamebalanceconcepts.wordpress. com (17.09.25)

5. *Becker A., Görlich D.* What is game balancing? An examination of concepts // ParadigmPlus. 2020. Vol. 1, No. 1. P. 22–41.

6. *Andrade G. et al.* Dynamic game balancing: An evaluation of user satisfaction // In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2006. Vol. 2, No. 1. P. 3–8.

7. *Gonçalves D. et al.* The Trick is to Stay Behind?: Defining and Exploring the Design Space of Player Balancing Mechanics // In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. 2024. P. 1–16.

8. *Sahibgareeva G.F., Kugurakova V.V., Bolshakov E.S.* Game Balance Tools // Russian Digital Libraries Journal. 2023. Vol. 26, No. 2. P. 225–251.

9. *Galimzyanov G.R.* Development of a tool for automatic adjustment of in-game parameters [Razrabotka instrumenta avtomaticheskoĭ korrektirovki vnutriigrovyh parametrov]: graduation. qualification. work on Bachelor, specialty Software Engineering. 2021. Kazan Federal University. 35 p. (In Russ.)

СВЕДЕНИЯ ОБ АВТОРАХ

РАХМАНКУЛОВА Валерия Рашидовна – выпускник бакалавриата Института ИТИС КФУ.

Valeria Rashidovna RAKHMANKULOVA – graduate at the Department of Software Engineering of the Institute of ITIS KFU.

email: raxmankulova.v@mail.ru

ORCID: 0009-0006-8425-616X

КУГУРАКОВА Влада Владимировна – доцент, кандидат технических наук, зав. кафедрой индустрии разработки видеоигр Института ИТИС КФУ.

Vlada Vladimirovna KUGURAKOVA, Ph.D. of Engineering Sciences, Head of the Video Game Development Industry Department of ITIS KFU.

email: vlada.kugurakova@gmail.com

ORCID: 0000-0002-1552-4910

Материал поступил в редакцию 1 сентября 2025 года