УДК 004.9
DOI 10.17513/snt.40112

РАЗРАБОТКА ИГРОВЫХ МЕХАНИК КОМПЬЮТЕРНОЙ ИГРЫ В ЖАНРЕ «ГОЛОВОЛОМКА» НА ПЛАТФОРМЕ UNITY

Киргизова Е.В., Фирер А.В.

Лесосибирский педагогический институт – филиал ФГАОУ ВО «Сибирский федеральный университет», Лесосибирск, e-mail: fivr@yandex.ru

Цель данного исследования – описать разработку игровых механик 3D игры в жанре «головоломка» на платформе Unity. Исследование рассматривает проблему разработки игр, такую как необходимость переписывать значительную часть кода при создании новых игр, несмотря на использование схожих механик и компонентов. Предлагается компонентно-ориентированная модель разработки, которая позволяет разбивать проект на конкретные компоненты для облегчения их повторного использования. В статье подробно описана реализация ключевых игровых механик, включая клонирование, лазеры, активаторы, барьеры и др. Особое внимание уделено порталам, которые обогащают игровой процесс и предоставляют разнообразные возможности для решения головоломок. В рамках исследования выявлен ряд принципов телепортации. Для реализации телепортации объектов и игроков в компьютерной игре авторы предлагают использовать несколько методов, таких как PlayerTeleport (предназначен для корректной телепортации игрока из одной точки в другую) и CloneObject (предназначен для реализации эффекта непрерывного физического перемещения). Для разработки визуальной составляющей порталов использованы методы Start и CameraPortal. Результаты показывают, что внедрение этих механик улучшает игровой опыт, предоставляя игрокам широкий спектр для экспериментов и поиска нестандартных решений.

Ключевые слова: компьютерная игра, игровой движок, жанр «головоломка», игровые механики

DEVELOPMENT OF GAME MECHANICS OF A COMPUTER GAME IN THE PUZZLE GENRE ON THE UNITY PLATFORM

Kirgizova E.V., Firer A.V.

Lesosibirsk Pedagogical Institute – branch of the Siberian Federal University, Lesosibirsk, e-mail: fivr@yandex.ru

The purpose of this study is to describe the development of game mechanics of a 3D puzzle game on the Unity platform. The study examines the problems of game development, such as the need to rewrite a significant part of the code when creating new games, despite the use of similar mechanics and components. A component-based development model is proposed, which allows you to break a project into specific components to facilitate their reuse. The article describes in detail the implementation of key game mechanics, including cloning, lasers, activators, barriers and others. Special attention is paid to portals that enrich the gameplay and provide a variety of opportunities to solve puzzles. The study revealed a number of principles of teleportation. To implement teleportation of objects and players in a computer game, the authors suggest using several methods, such as PlayerTeleport (designed to correctly teleport a player from one point to another) and CloneObject (designed to implement the effect of continuous physical movement). The Start and CameraPortal methods were used to develop the visual component of the portals. The results show that the implementation of these mechanics improves the gaming experience, providing players with a wide range of experiments and finding non-standard solutions.

Keywords: computer game, game engine, puzzle genre, game mechanics

Введение

Компьютерные игры в жанре головоломок на протяжении многих лет сохраняют свою популярность, привлекая игроков интеллектуальными вызовами, оригинальными механиками и возможностью развить логическое мышление. Современные технологии, в частности игровой движок *Unity*, предоставляют разработчикам мощные инструменты для создания интерактивных 3D-миров, открывая новые горизонты не только для жанра головоломок, но и для обучающих симуляторов, систем человеко-машинных коммуникаций и т.п. Этим можно объяснить интерес исследователей и разработчиков к среде Unity. Так, Р.А. Карелова и П.С. Коробейников [1] выделяют наиболее часто встречающиеся проблемы начинающих разработчиков, связанные с контролем над проектом, и предлагают пути их решения; В.А. Зенг [2] рассматривает процесс реализации прототипа бесконтактного взаимодействия пользователя с системой с помощью игрового движка Unity 3D; С.Д. Лизяев и Р.С. Молотов [3] делают акцент на особенностях создания и редактирования анимации при разработке имитационных моделей транспортных средств в среде Unity.

Игра *Portal 2*, выпущенная в 2011 году, стала настоящим прорывом в жанре голо-

воломок благодаря инновационной механике порталов и захватывающему сюжету. Игра продемонстрировала потенциал головоломок как основы для создания глубоких и эмоционально вовлекающих игровых опытов.

По мере развития индустрии видеоигр создавались различные высокобюджетные проекты, которые включали в себя несколько поджанров, чтобы добавить погружение в атмосферу и виртуальный мир. Так головоломки стали основой для увлекательного игрового опыта в самых крупных проектах. В настоящее время ни одна высокобюджетная компьютерная игра не обходится без элементов головоломок.

Цель исследования – описать разработку игровых механик 3D-игры в жанре «головоломка» на платформе Unity.

Материалы и методы исследования

Современная разработка игр, в том числе игр в жанре «головоломка», сталкивается с рядом проблем, связанных с архитектурой игровых проектов и необходимостью переписывать значительную часть кода при создании новых игр, даже если они используют схожие механики и компоненты. Это вызвано тем, что код часто оказывается специфичным для конкретного проекта и плохо поддается повторному использованию.

Чтобы уменьшить риск повторного переписывания компонента под определенную игру, существует компонентно-ориентированная модель разработки. Эта модель позволяет разбивать весь проект на конкретные компоненты, каждый из которых предназначен для выполнения определенной задачи. В этом случае важной проблемой при разработке игр является большое количество взаимосвязей между компонентами.

В книге «Проектирование и архитектура игр» Э. Роллингза и Д. Морриса [4, с. 688] представлена архитектура типовой игры в соответствии с рисунком 1, состоящая из различных систем.

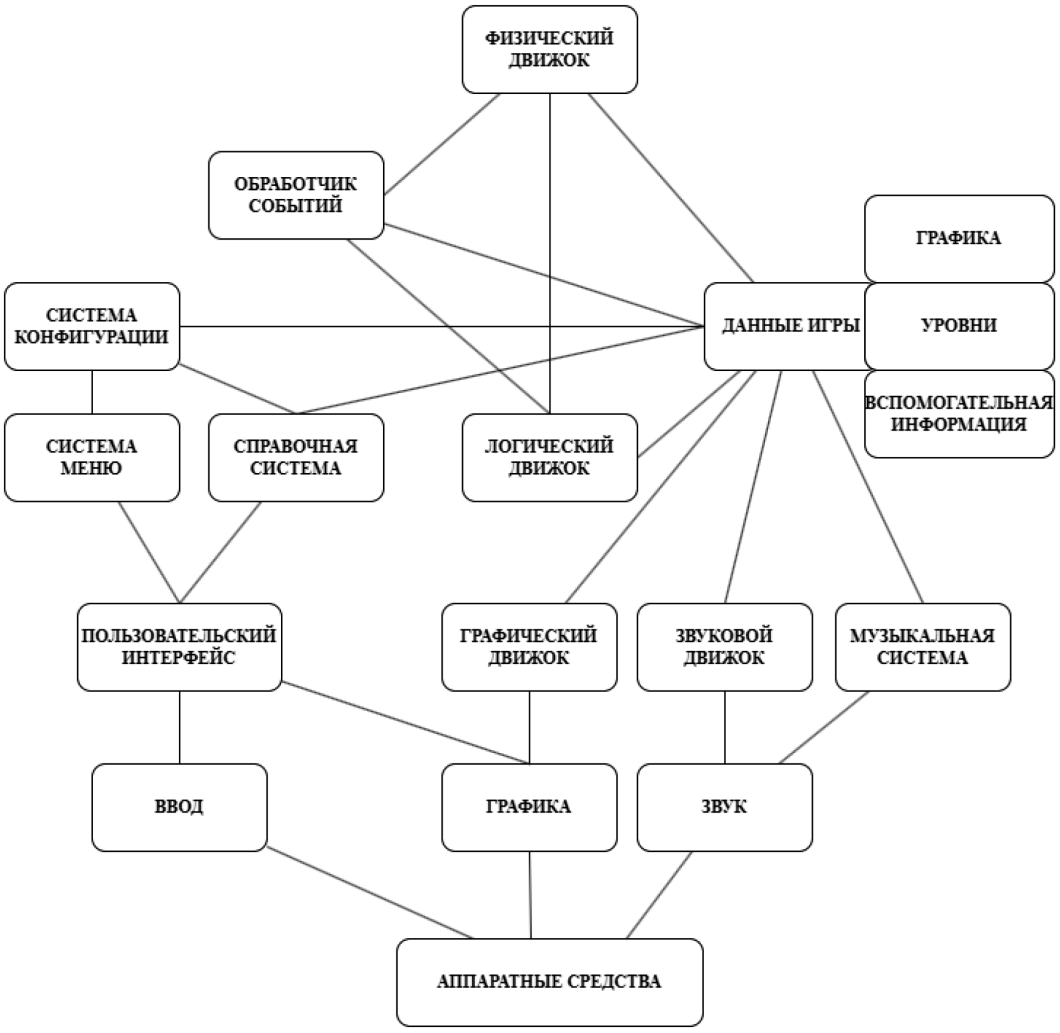


Рис. 1. Архитектура типовой игры

Каждая система может быть декомпозирована на большое количество компонентов, которые могут быть связаны как между собой, так и между другими системами и их компонентами. Такая модель архитектуры позволит создавать универсальные компоненты, которые можно будет применять во всех последующих проектах.

С учетом компонентно-ориентированной модели такая архитектура типовой игры позволит повысить эффективность, гибкость и расширяемость разрабатываемой системы.

Современная разработка игр должна опираться на основы игрового дизайна. Вслед за Р. Рауз под игровым дизайном будем понимать «… процесс создания формы и содержания игрового процесса (геймплея) разрабатываемой игры» [5, с. 660].

По мнению Н.Ю. Казаковой [6, с. 218], наиболее важным фактором успеха разработки компьютерных игр является наличие проработанного сюжета, способного заинтересовать пользователя и гармонично объединяющего визуальный ряд проекта с его игровыми механиками, которые представляются основными составляющими игрового дизайна.

«Под понятием игровые механики понимают совокупность действий, которые может совершить пользователь, а также совокупность правил и ограничений» [7, с. 23]. Все множество игровых механик игры формирует конкретную реализацию ее игрового процесса.

Результаты исследования и их обсуждение

Основой для игр в жанре головоломок являются ее игровые механики, определяющие правила взаимодействия игрока с миром и создающие основу для интеллектуальных вызовов. В разработанной игре реализованы следующие механики.

1. Клон – это способность игрока, которую он может использовать, когда захочет. Способность заключается в том, что игрок может создавать клона своего персонажа в указанном им месте с помощью курсорного указателя. Создание клона реализовано через префаб («образец» объекта с заранее настроенными компонентами) персонажа без компонентов управления и камеры. Создание и удаление производятся посредством нажатия на левую и правую кнопку мыши в определенной точке на экране, то есть игрок указывает курсором, где именно он хочет создать клона. Регистрация нажатий осуществлена через используемый инструмент InputSystem и mousePosition – это свойство системы ввода, которое передает текущее положение мыши в координатах.

2. Лазеры – представляют собой в игре не просто яркие лучи электрического света, а динамические элементы геймплея. При попытке пересечь луч лазера игрок будет откинут назад. Лазеры могут быть прерваны физическими объектами, такими как кубы, стены или специальные преграды. Это позволяет игроку манипулировать траекторией лазеров, направляя их в нужные точки или блокируя их воздействие. Клоны также могут прерывать лазеры, позволяя игроку создавать «живые» барьеры. Работа лазеров построена на компоненте *LineRenderer* в *Unity.*

3. Порталы – являются одной из ключевых механик игры, предлагающей игроку уникальную возможность мгновенного перемещения в пространстве и манипуляции с объектами и лазерными лучами. В комнатах могут находиться два статических портала, заранее размещенных в определенных точках. При входе в один из порталов игрок мгновенно перемещается в другой портал, расположенный в той же комнате. Игрок может переносить через порталы объекты, которые он берет в руки. Лазерные лучи также могут проходить через порталы, продолжая свою траекторию с другой стороны. Если в игре есть перемещение, то добавление портала создаст огромное количество вариаций уровней или решений этих уровней.

В рамках исследования выявлен ряд принципов телепортации (с англ. *teleportation* – «перемещать на расстоянии») объектов:

− перемещение объектов из одной точки в другую (в случае перемещения объекта через портал);

− реализация эффекта непрерывного перемещения через порталы (например, персонаж проходит через дверь или окно);

− реализация визуальной части портала (чтобы игрок видел, что находится перед выходом из портала).

Для реализации телепортации объектов и игроков в компьютерной игре используются несколько методов. Остановимся на них подробнее.

Метод *PlayerTeleport* – предназначен для корректной телепортации игрока из одной точки в другую, на основании него:

а) вычисляется вектор *a* от портала (*Portal*) до игрока (*Player*) по формуле $a = Player - Portal$, для того чтобы определить, где находится объект по отношению к порталу в соответствии с рисунком 2;

б) для того, чтобы определить местоположение игрока на рисунке относительно портала (игрок перед порталом и внутри портала), находится скалярное произведение *Dot* вектора портала (B_1) и вектора (нормали) от портала до игрока (A_1, A_2), в соответствии с таблицей 1. Если $Dot > 0$, то игрок зашел в портал, иначе игрок находится перед порталом;

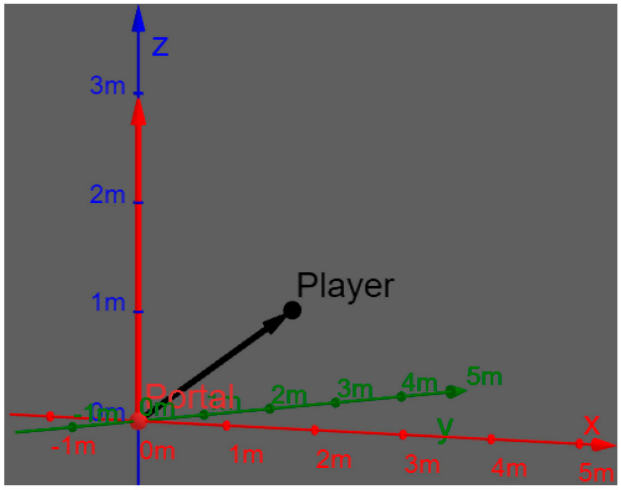


Рис. 2. График вектора портала и игрока

Таблица 1

Определение позиции игрока относительно портала

Случай, когда игрок находится перед порталом	Случай, когда игрок находится внутри портала
$A_1 = Player - Portal = \begin{pmatrix}1\\1\\1\end{pmatrix} - \begin{pmatrix}0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\1\\1\end{pmatrix}$	$A_2 = Player - Portal = \begin{pmatrix}1\\1\\-1\end{pmatrix} - \begin{pmatrix}0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\1\\-1\end{pmatrix}$
$B_1 = \begin{pmatrix}0\\0\\3\end{pmatrix}$	$B_1 = \begin{pmatrix}0\\0\\3\end{pmatrix}$
$Dot = A_1 * B_1 = \begin{pmatrix}1\\1\\1\end{pmatrix} * \begin{pmatrix}0\\0\\3\end{pmatrix} = 3$	$Dot = A_2 * B_1 = \begin{pmatrix}1\\1\\-1\end{pmatrix} * \begin{pmatrix}0\\0\\3\end{pmatrix} = -3$

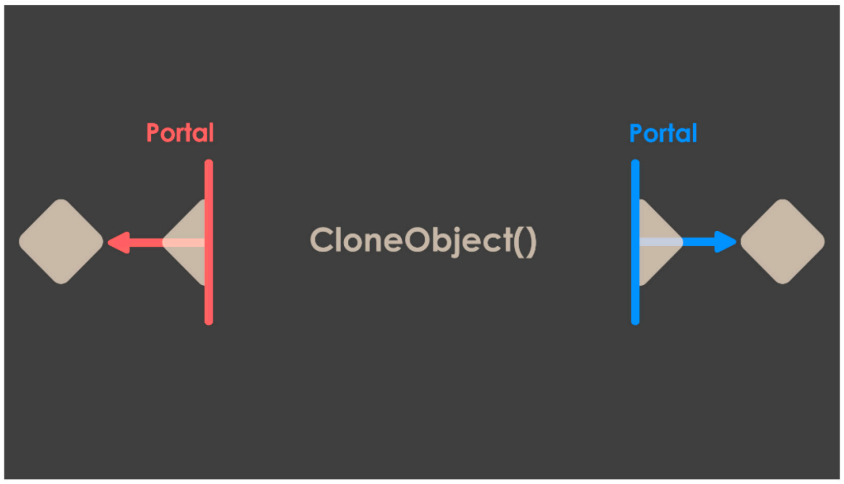


Рис. 3. Схема работы метода CloneObject

в) для корректной телепортации игрока относительно выхода вычисляется угол поворота игрока к входу по формуле: *rotationDiff* = – *Quaternion.Angle*(Q_1, Q_2) + 180° (где Q_1 –ориентация первого портала, Q_2 – ориентация второго портала), и находится разница между ориентациями первого и второго порталов. Используемая функция *Quaternion.Angle* позволяет вычислить угол между двумя кватернионами (ориентациями). При вычислении угла поворота игрока к входу добавляется 180º для того, чтобы учесть различия ориентаций порталов. Далее находится смещение позиции игрока к вектору портала (*portalToPlayer*) после поворота с использованием функции *Quaternion.Euler*.

Для реализации эффекта непрерывного физического перемещения применяется метод *CloneObject*, схема работы которого представлена на рисунке 3.

Для создания клона необходимо выполнить следующее:

а) при создании клона объекта (всех кроме лазера) присваивается оригинальный угол поворота и назначается позиция портала, из которого будет выходить объект, соответствующий его клону;

б) при создании клона объекта лазера присваиваются оригинальный угол поворота и позиция начальной точки, которая переводится в локальные координаты портала. Затем создается новая позиция начальной точки, полученный поворот и начальная точка переводятся в мировые координаты дочернего портала.

На основе документации *Unity* [8] определяются методы, используемые для визуальной части:

– метод *Start* – создает текстуру рендера для камеры дочернего портала и назначает ее как текстуру материала родительского портала, чтобы отображать изображение, видимое через дочерний портал;

– метод *CameraPortal* – выполняет следующие вычисления корректного перемещения камер порталов, которые передают изображение, в соответствии с таблицей 2.

4. Активаторы дверей/механизмов – это базовые кнопки, которые могут открывать проход куда-либо, только визуально выглядят по-другому. Один из таких механизмов активируется, когда внутрь помещается какой-то физический объект (куб или клон). Другой активируется, когда лазер коснется его.

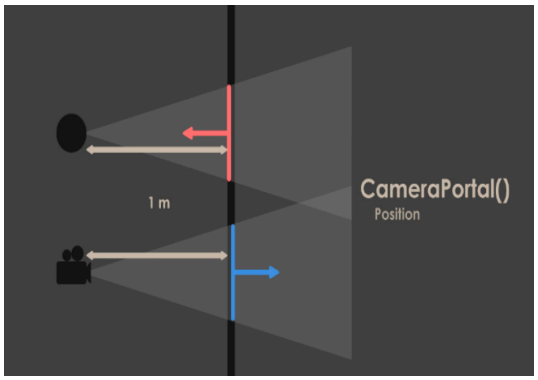
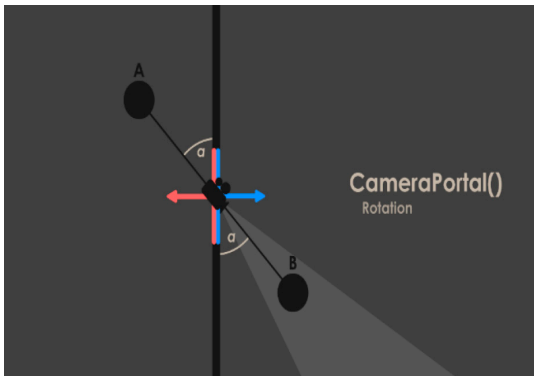
Первый тип активаторов работает, когда в него помещают физический объект. В этом случае для создания области взаимодействия используется *Trigger* (свойство компонента коллайдера, который делает объект областью, способной взаимодействовать с входящими в нее объектами). С помощью кода создается работа этой зоны, представленная на рисунке 4. Когда объект помещается в зону, то он активирует механизм (например, открытие двери) и притягивается к центру этой зоны.

Второй тип активаторов работает от лазера. Как и в первом варианте активатора, используется *Trigger*. Когда попадает лазер, активируется механизм, когда лазер выходит из триггера, механизм деактивируется. Схематично компонент представлен на рисунке 5.

Компоненты *TriggerAreaController* и *LaserReciever* работают по одинаковому принципу: когда в их коллайдер попадает объект, они активируются.

Таблица 2

Применение метода CameraPortal

Нахождение позиции игрока в локальных координатах первого портала и присвоение этих координат камере второго портала	Нахождение разницы в повороте между текущим порталом и дочерним порталом и корректировка поворота камеры портала
	

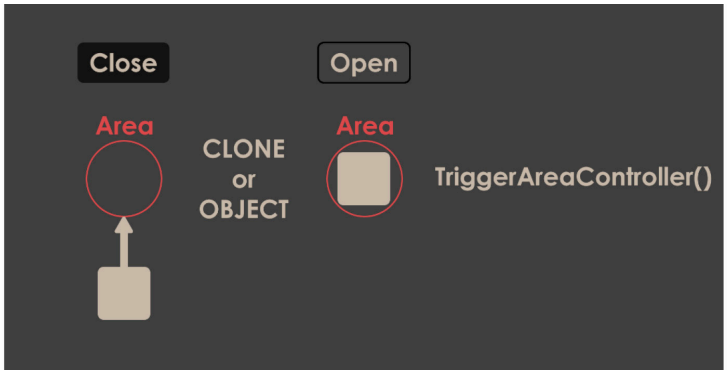


Рис. 4. Схема работы метода TriggerAreaController

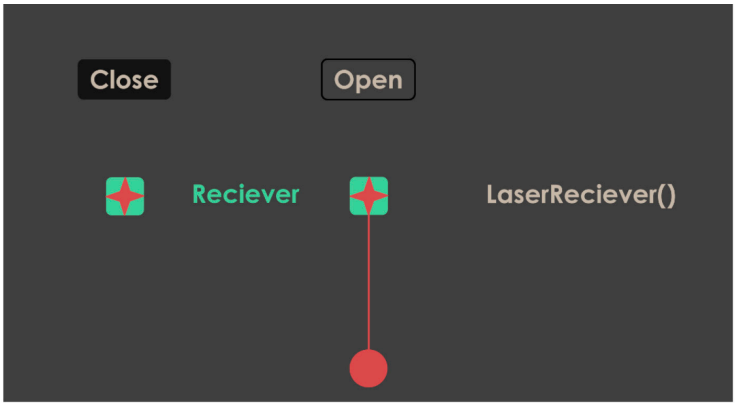


Рис. 5. Схема работы метода LaserReciever

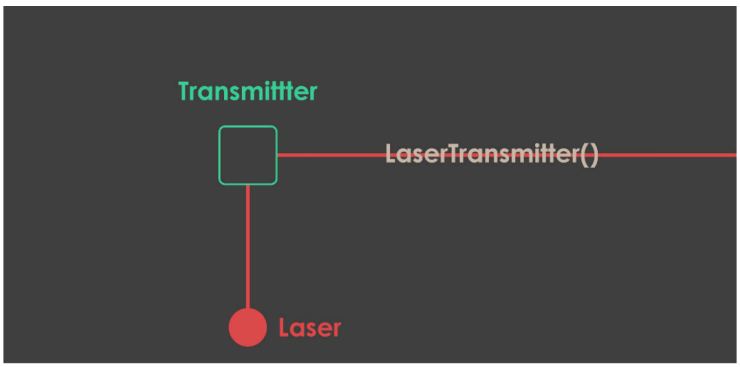


Рис. 6. Схема работы метода LaserTransmitter

5. Передатчик лазера, внешне напоминающий обычный куб, представленный на рисунке 6, обладает способностью перенаправлять лазерный луч. Попадания лазера обрабатываются с помощью столкновения (*Collision*) двух коллайдеров. При попадании лазером в объект передатчика компонент внутри лазера создает клон лазера в заранее определенной точке с одной из сторон.

При попадании лазера в объект передатчика компонент *LaserTransmitter* создает клон лазера в заранее определенной точке (*point*) с одной из сторон передатчика. Это позволяет игроку легко менять направление лазера, взяв в руки объект передатчика.

6. Барьеры – это электрические поля, которые:

– не пропускают игрока и никак не влияют на движение других объектов или лазера;

– пропускают только игрока.

Барьеры в игре работают через столкновения коллайдеров и свойство компонента *Rigidbody* – *ExcludeLayers* (в переводе на русский – «исключенные слои»).

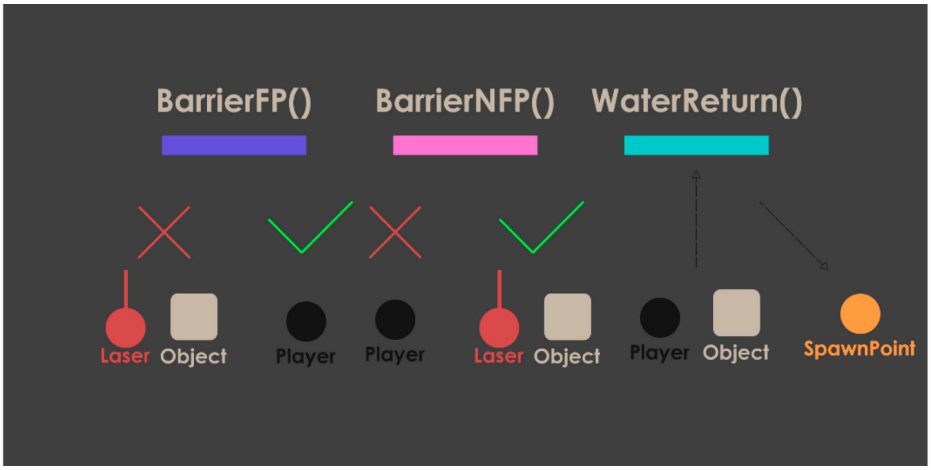


Рис. 7. Схема работы компонентов барьеров

Работа барьеров, представленная на рисунке 7, разделяется на следующие виды:

– для первого барьера, который пропускает только игрока, в созданном компоненте составлено условие «если игрок пытается пройти через него с объектом в руках, или если объект сам проходит через барьер, объект автоматически выбрасывается из рук игрока»;

– для второго барьера, который пропускает все объекты, кроме игрока, в созданном компоненте составлено условие «когда у персонажа в руках находится объект и он задевает этот барьер, то объект выбрасывается из его рук»;

– для третьего типа барьеров жидкости в пропастях создан компонент, который возвращает в начало головоломки игрока или объекты, попавшие в эту пропасть.

Заключение

Реализация основных механик взаимодействия с окружающим миром, таких как телепортация, прохождение через барьеры и взаимодействие с лазерами, обогащает игровой процесс и создает основу для разнообразных головоломок. Если в игре планируется перемещение, то добавление портала создаст огромное количество вариаций уровней или решений этих уровней. При разработке следует придерживаться ряда принципов телепортации. Для корректной и визуально привлекательной телепортации объектов и игроков в компьютерной игре рекомендуется использовать несколько методов, таких как PlayerTeleport, CloneObject, Start и CameraPortal.

Сочетание разнообразных игровых механик с уникальной способностью клонирования персонажа открывает перед игроком широкие возможности для экспериментирования и поиска нестандартных решений.

Список литературы

1. Карелова Р.А., Коробейников П.С. Контроль над проектом на Unity: частые проблемы начинающих разработчиков и пути их решения // Международный научно-исследовательский журнал. 2020. № 5-1 (95). С. 40-45. DOI: 10.23670/IRJ.2020.95.5.006.

2. Зенг В.А. Создание прототипа компьютерного бесконтактного компьютерного интерфейса в Unity 3D // Известия Тульского государственного университета. Технические науки. 2019. № 12. С. 480-485.

3. Лизяев С.Д., Молотов Р.С. Особенности создания анимации при разработке обучающих симуляторов в среде Unity // Вестник Ульяновского государственного технического университета. 2016. № 3 (75). С. 41–43.

4. Роллингз Э. Проектирование и архитектура игр. 2-е изд. М.: Вильямс, 2006. 698 с.

5. Рауз Р. Дизайн игры: теория и практика. 2-е изд. Пано: Wordware Publishing, 2004. 698 с.

6. Казакова Н. Ю. Основные принципы разработки сюжета игрового проекта в рамках гейм-дизайна // Вестник Адыгейского государственного университета. Серия 2: филология и искусствоведение. 2016. № 3 (182). С. 216-222.

7. Логинов К.В. Метод управления процессом прохождения учебного курса с применением событийно-ориентированных игровых механик: дис. … канд. техн. наук. Санкт-Петербург, 2021. 169 с.

8. Документация Unity. [Электронный ресурс]. URL: https://docs.unity3d.com (дата обращения: 04.06.2024).