8Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный университет»

Школа развития цифровых компетенций «Digital Up» (цифровая кафедра)

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**на итоговый проект «Разработка головоломки BlockFlow»**

**по ДПП ПП «Основы Gamedev и VR-разработки»**

|  |  |  |  |
| --- | --- | --- | --- |
| п/п  № | Задание | Исполнитель | Рабочий график (план) выполнения |
| 1 | Исследовать логические основы построения уровней в игре BlockFlow и аналогичных головоломках. | Лазутина А.С.  Нербышева В.Г. | 09.06.2025 – 11.06.2025 |
| 2 | Разработать собственный набор уровней с возрастающей сложностью. | 12.06.2025 – 19.06.2025 |
| 3 | Протестировать уровни и сделать работу над ошибками. | 20.06.2025 – 22.06.2025 |
| 4 | Подготовка отчета и видео-демонстрации | 23.06.2025 – 24.06.2025 |

Руководитель проекта   
старший преподаватель кафедры культурологии и дизайна Каратаев А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / «\_\_\_ » \_\_\_\_\_\_\_\_\_ 2025 г.

(подпись)

СОГЛАСОВАНО:

Руководитель ДПП ПП

канд. физ.-мат. наук, доцент Козлов Д.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2025 г.

(подпись)

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный университет»

Школа развития цифровых компетенций «Digital Up» (цифровая кафедра)

Отчет о выполнении группового итогового проекта по ДПП ПП

«Основы Gamedev и VR-разработки»

**«Разработка головоломки BlockFlow»**

Исполнители:

Лазутина А.С.  
Нербышева В.Г.

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 г.

Руководитель проекта

старший преподаватель кафедры культурологии и дизайна Каратаев А.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 г.

г. Барнаул, 2025

# Цель проекта

Цель данного проекта разработка головоломку BlockFlow.

# Задачи проекта и исполнители

Для достижения поставленной цели требовалось решить ряд задач:

1. Исследовать логические основы построения уровней в игре BlockFlow и аналогичных головоломках.
2. Разработать собственный набор уровней с возрастающей сложностью.
3. Протестировать уровни и сделать работу над ошибками.

# Актуальность и востребованность проекта

В настоящее время логические головоломки, особенно мобильного формата, продолжают пользоваться высоким спросом среди широкой аудитории пользователей благодаря своей доступности, компактности и способности развивать критическое мышление и стратегическое планирование. Игра "BlockFlow" актуальна в контексте растущего интереса к интеллектуальному досугу, когда игроки стремятся не просто развлечься, а получить когнитивную нагрузку, способствующую развитию логики, внимания и пространственного мышления. На фоне перенасыщенности рынка однотипными аркадами и казуальными проектами, головоломки с продуманной механикой и прогрессивной сложностью остаются востребованными за счет своей способности удерживать внимание на длительное время, предлагая игроку чувство удовлетворения от решения сложных задач. Уникальные элементы, такие как направленные движения блоков, фиксированные препятствия позволяют игре выделяться среди аналогов, предлагая свежий взгляд на классическую механику. Кроме того, "BlockFlow" легко масштабируется, позволяя расширять контент за счет добавления новых уровней, элементов и условий, что делает её привлекательной как для индивидуальных пользователей, так и для разработчиков, заинтересованных в сотрудничестве или адаптации продукта под образовательные или развлекательные платформы. Всё это подчеркивает как актуальность проекта в современных реалиях, так и его коммерческий и образовательный потенциал.

# Общие сведения о проделанной работе

Этапы выполнения проекта:

* Исследовать логические основы построения уровней в игре BlockFlow и аналогичных головоломках.
* Разработать собственный набор уровней с возрастающей сложностью.
* Протестировать уровни и сделать работу над ошибками.

Проект по созданию логической игры BlockFlow начался с тщательной проработки концепции, анализа жанра и постановки задач, направленных на формирование целостного, логически обоснованного игрового процесса. BlockFlow — это 3D-головоломка, в которой игрок должен перемещать блоки в ограниченном пространстве, чтобы освободить путь для целевого блока и вывести его через заданный выход. Игра вдохновлена известными представителями жанра, такими как *Unblock Me*, *Block Escape*, *Move the Block: Slide Puzzle*, *Pull the Pin*, *Parking Jam 3D* и классическая *Rush Hour*. Каждая из этих игр дала ценные идеи для построения уровней, создания сбалансированной системы прогрессии и разработки интересных механик.

На первом этапе был проведён анализ существующих решений и реализованных механик в аналогичных играх. Было замечено, что успешные проекты такого рода часто строятся по принципу «от простого к сложному», где сложность повышается не только за счёт размера поля и количества блоков, но и за счёт добавления интерактивных элементов, таких как телепорты, переключатели, зафиксированные объекты и преграды.

В качестве основного языка программирования был выбран **C#**, а для создания игрового прототипа и полной реализации игрового процесса — профессиональный движок **Unity**. Данный выбор обусловлен тем, что Unity предоставляет широкий набор инструментов для 2D и 3D-разработки, имеет отличную документацию и обширное сообщество, а также полностью интегрирован с C#. Средой разработки стала **Visual Studio**, которая обеспечивает высокую производительность, гибкую отладку и стабильную работу с Unity.

Разработка началась с создания пользовательского интерфейса и главного меню (см. скрипт MainMenu.cs). Игрок может начать новую игру, выйти из приложения, включить или отключить музыку, а также вернуться к последнему пройденному уровню. В процессе игры при переходе в меню используется система PlayerPrefs для запоминания текущего уровня, на котором находится игрок. Это позволяет, например, при отключении звука или выполнении других действий в меню не терять прогресс: игра сохраняет информацию о текущем уровне и не возвращает игрока на первый, а сохраняет его на достигнутом этапе. Метод ReturnToLastLevel() позволяет в любой момент вернуться к сохранённому уровню, обеспечивая удобство и непрерывность игрового процесса.

Следующим шагом стало создание прототипа игрового уровня, в котором игрок управляет блоками с помощью клавиш. В Player.cs реализована логика движения игрового объекта по осям в определённом направлении, которое заранее задается через переменную moveDirection. Управление реализовано с помощью KeyCode, что дает гибкость в настройке: можно задавать уникальные комбинации клавиш для разных блоков. Движение сопровождается звуковым эффектом, реализованным с помощью AudioSource и AudioClip, что повышает вовлеченность игрока.

Особое внимание было уделено системе взаимодействия объектов. В скрипте Activator.cs реализованы логические переключатели, реагирующие на вход и выход определенных объектов (блоков с тегом Cube или игрока с тегом Player). В зависимости от того, кто и в какой момент касается триггерной зоны, изменяется состояние групп объектов — они либо становятся проходимыми (используя isTrigger = true), либо непроходимыми, а их визуальное представление меняется через материалы (normal и transparent). Таким образом реализуются механики скрытых и временно отключаемых препятствий, что делает игровой процесс динамичным и непредсказуемым.

Также в Player.cs была добавлена базовая логика перехода между уровнями. Как только игрок доходит до блока с тегом Finish, автоматически происходит загрузка следующего уровня. Это реализовано через SceneManager.LoadScene(), что обеспечивает линейное прохождение.

Основной акцент в проекте делался на простоту взаимодействия и интуитивное управление. Было решено отказаться от избыточных меню и инструкций — вся необходимая информация подаётся через визуальные элементы и первые уровни, выступающие в роли обучающих. Такой подход заимствован у игр *Unblock Me* и *Move the Block*, где обучение встроено в геймплей.

Также было важно обеспечить модульность кода — все основные функции разделены по отдельным скриптам. Это упрощает отладку и масштабирование проекта. К примеру, реализация движений в Player.cs может быть повторно использована для других типов блоков, просто изменив направление движения или клавиши управления.

Отказ от сложных математических расчетов и тяжёлой физики позволил добиться высокой производительности даже на слабых устройствах. Использование встроенного физического движка Unity в сочетании с настройкой Rigidbody дало необходимый уровень реалистичности без лишних затрат ресурсов.

В процессе разработки проводилось пошаговое тестирование каждого добавленного элемента. Были протестированы уровни различной сложности, выявлены ошибки в логике переключателей и моменты, где игрок мог непреднамеренно застревать между объектами. После анализа этих проблем код был переработан: улучшена система коллизий, добавлены ограничения на движение при соприкосновении с неподвижными объектами, а также реализована перезагрузка уровня по нажатию клавиши R.

Отдельно стоит отметить работу с пользовательским опытом. Например, в главном меню предусмотрена возможность отключения музыки — это улучшает комфорт игры для пользователей, предпочитающих тишину. Состояние сохраняется между сессиями, благодаря использованию PlayerPrefs.

# Результаты проекта

В ходе выполнения проекта было проведено большое количество испытаний. Так, например, на рисунках 1-5 представлены первые пять уровней до первого тестирования. Во время первого тестирования было замечено, что на уровне №4 выпадает один кубик-преграда, после чего было решено устранить ошибку, добавить еще 3 дополнительных уровня и усложнить предыдущие.

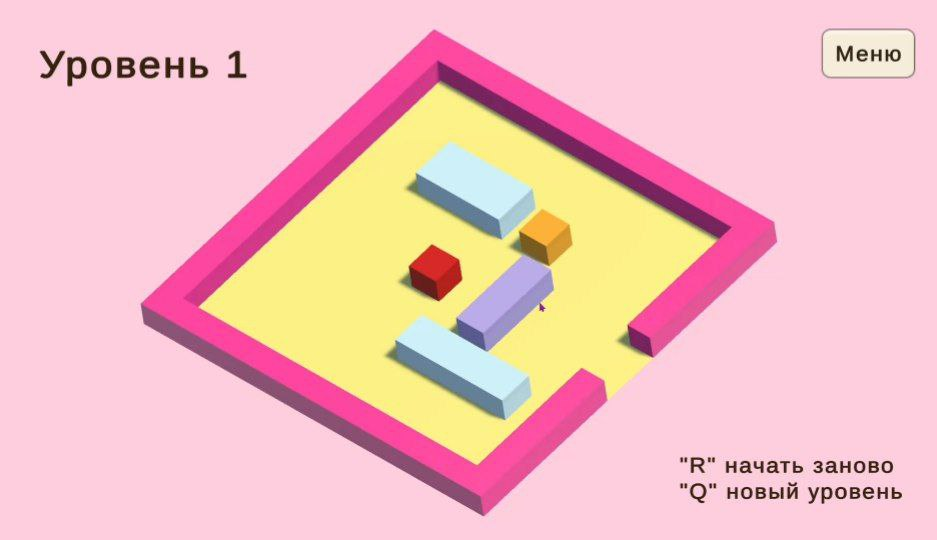
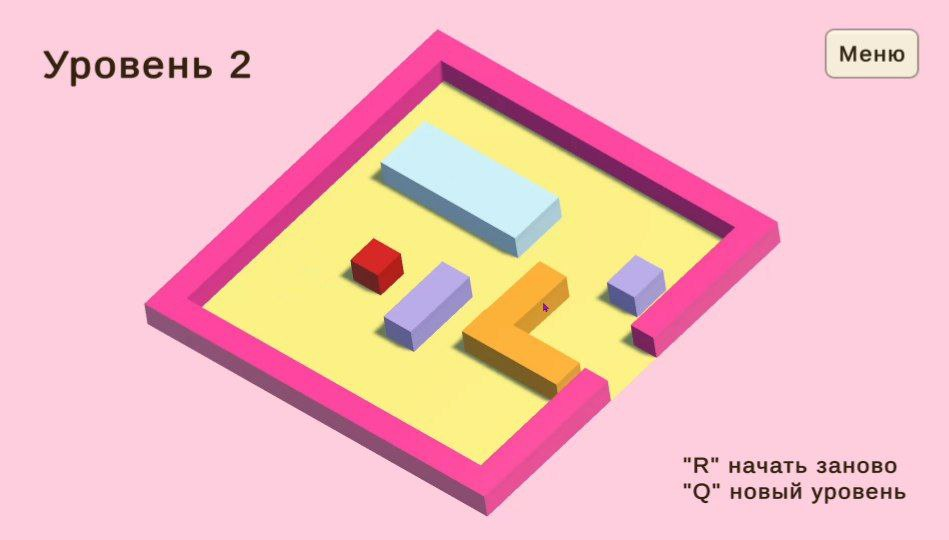


Рисунок 1.

Рисунок 2.

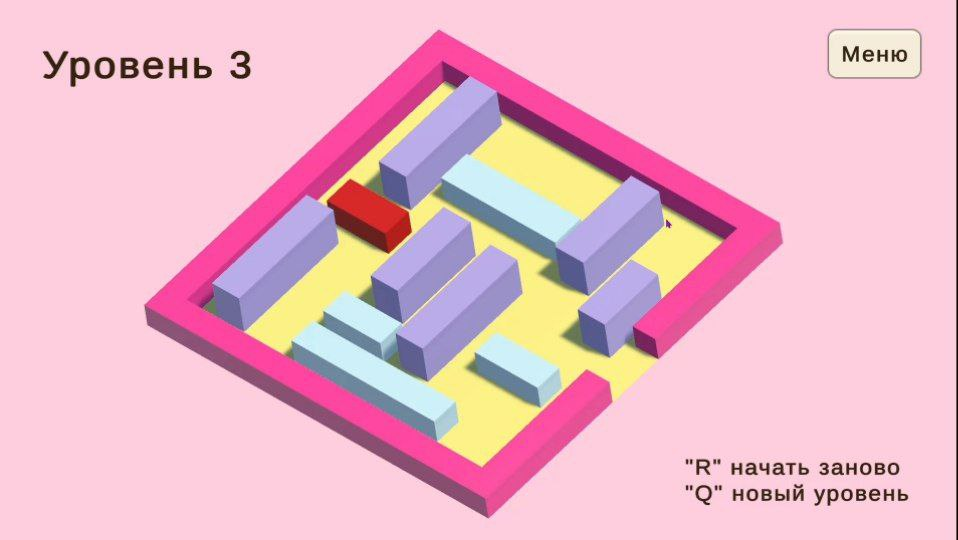


Рисунок 3.



Рисунок 4.

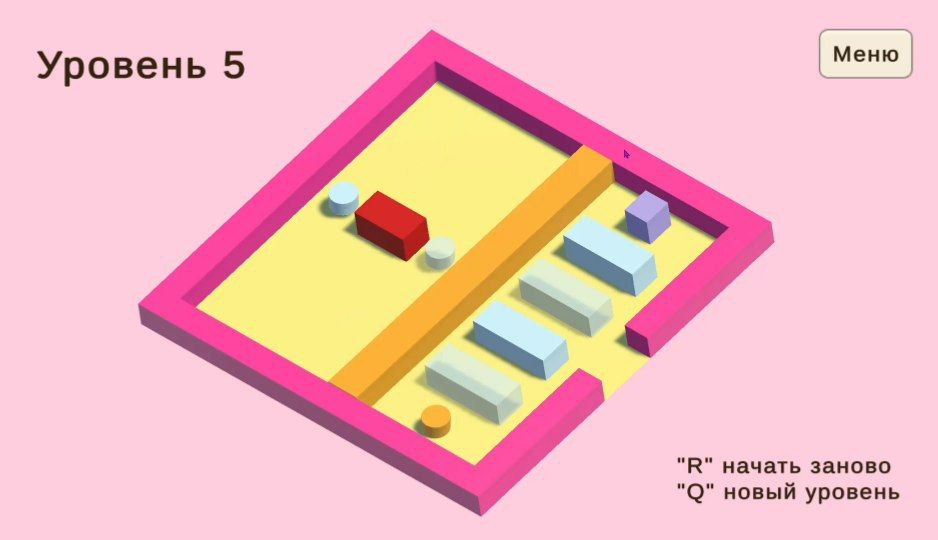


Рисунок 5.

В ходе тестирования уровней в игре **BlockFlow** были получены важные наблюдения, касающиеся сложности головоломок и реакции игрока на разные механики. Некоторые уровни, несмотря на простую на первый взгляд структуру, оказывались особенно сложными из-за неочевидного порядка перемещения блоков. Были случаи, когда даже несколько последовательных попыток не приводили к успешному завершению уровня, что указывает на необходимость дополнительной доработки структуры головоломки или добавления вспомогательных визуальных подсказок. В то же время, ряд уровней игроки проходили практически сразу, интуитивно понимая заложенную логику. Это позволило выделить закономерность: сложность должна возрастать плавно, чтобы игрок адаптировался к механике, а не сталкивался с резким скачком в трудности.

На основе проведённого анализа и тестирования можно сделать вывод о том, что создание качественной логической игры требует тщательной проработки не только визуальной составляющей, но и логики построения уровней. Игра **BlockFlow** демонстрирует, что с помощью простых, но продуманных механик можно создать глубокий и увлекательный игровой процесс, стимулирующий логическое мышление и пространственное восприятие. Полученный опыт может быть использован в будущих проектах, включая более масштабные головоломки, а также образовательные приложения, развивающие когнитивные способности игроков.

**Приложение 1.**



Рисунок 6 (Меню)

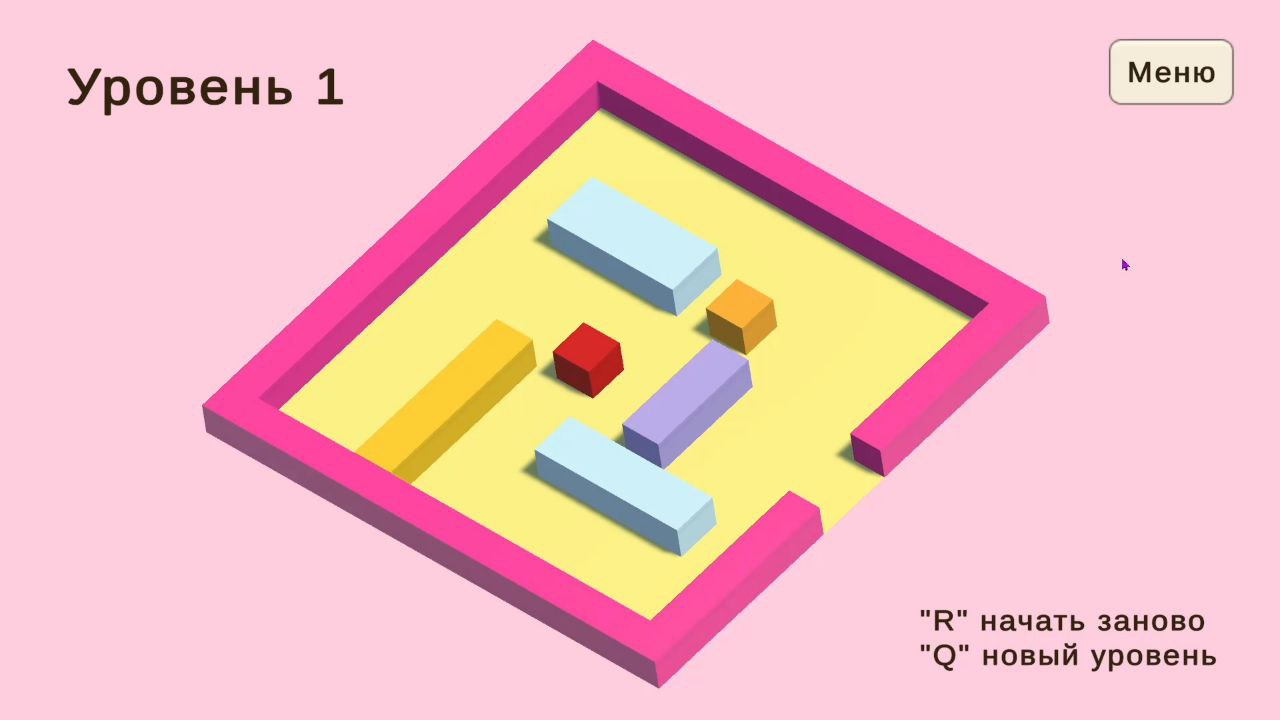


Рисунок 7 (Уровень 1)

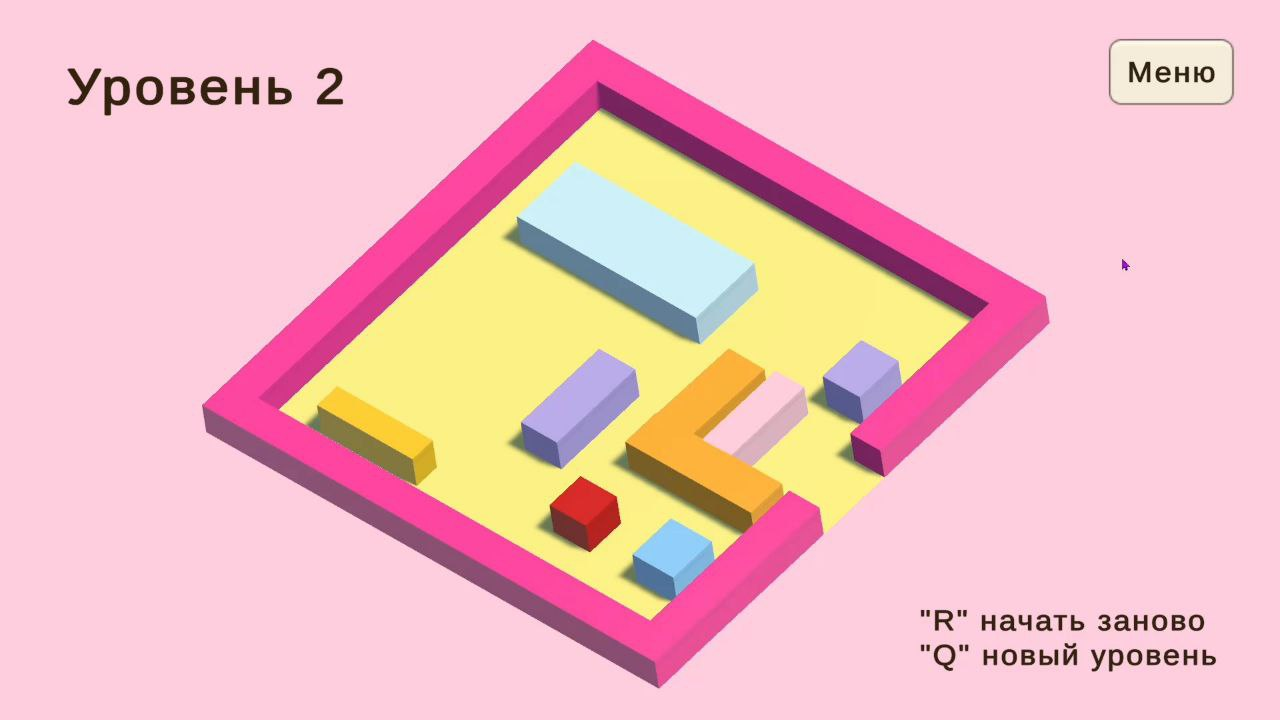


Рисунок 8 (Уровень 2)

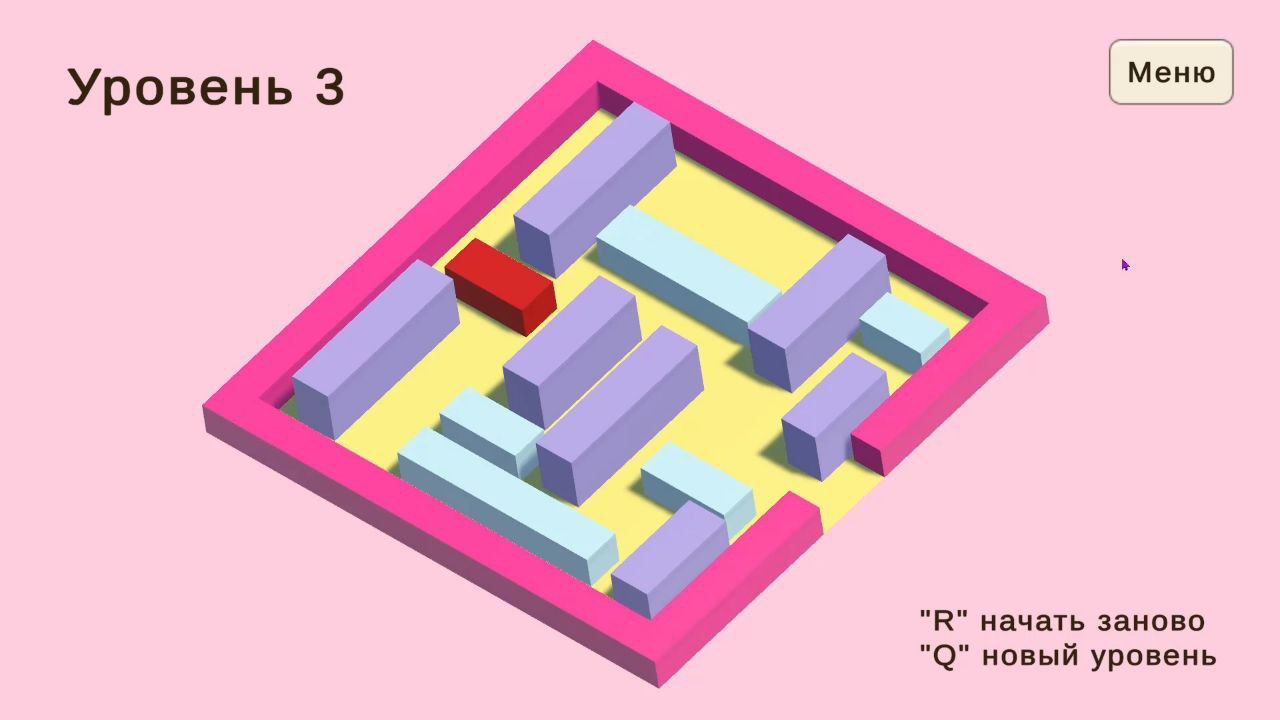


Рисунок 9 (Уровень 3)

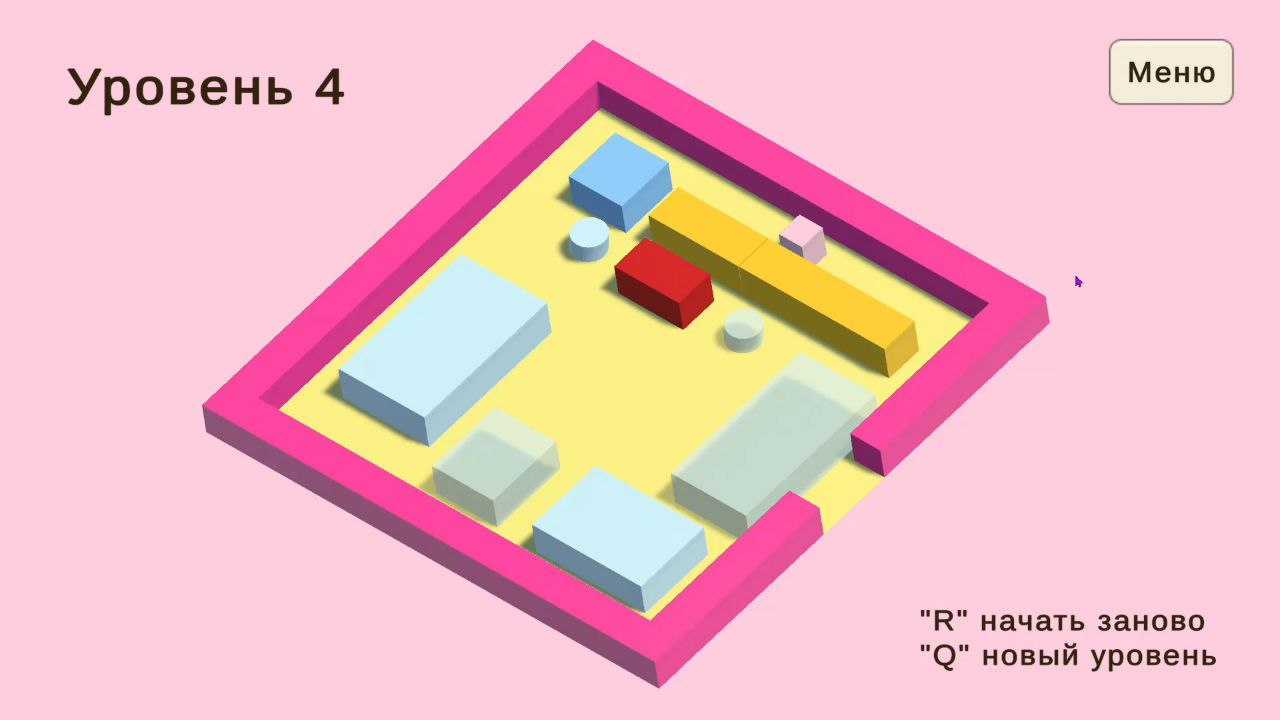


Рисунок 10 (Уровень 4)

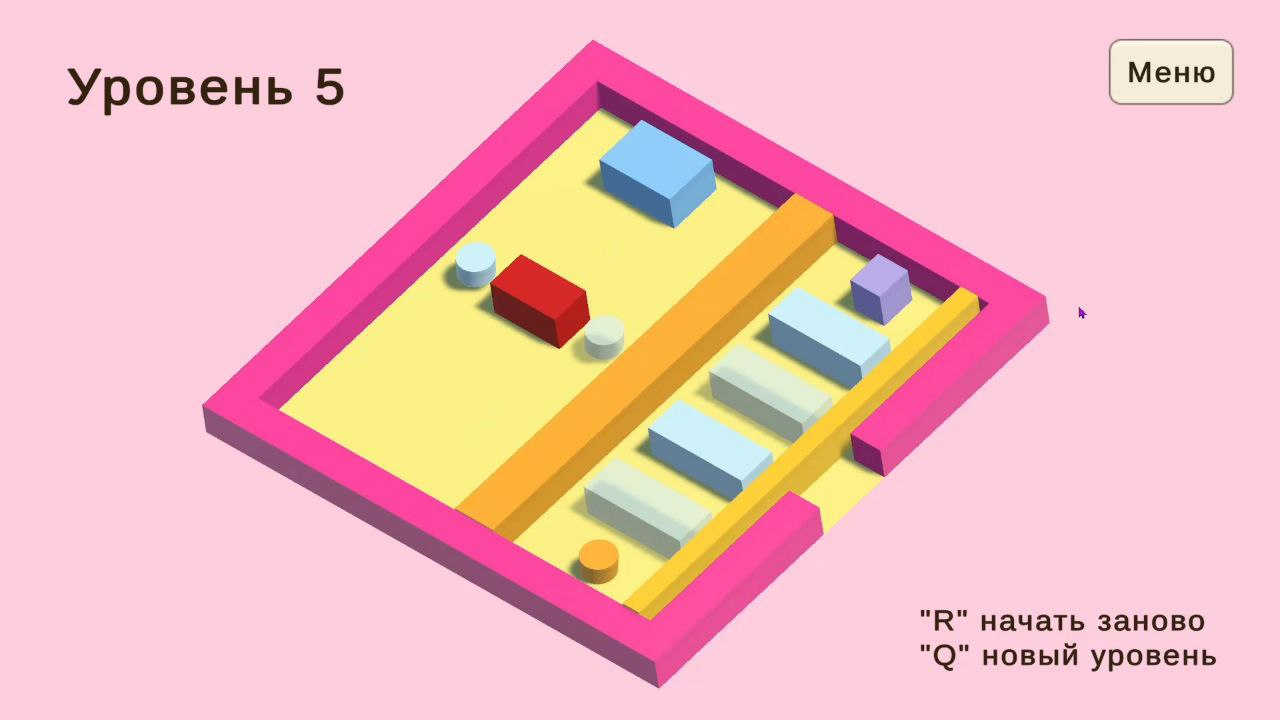


Рисунок 11 (Уровень 5)

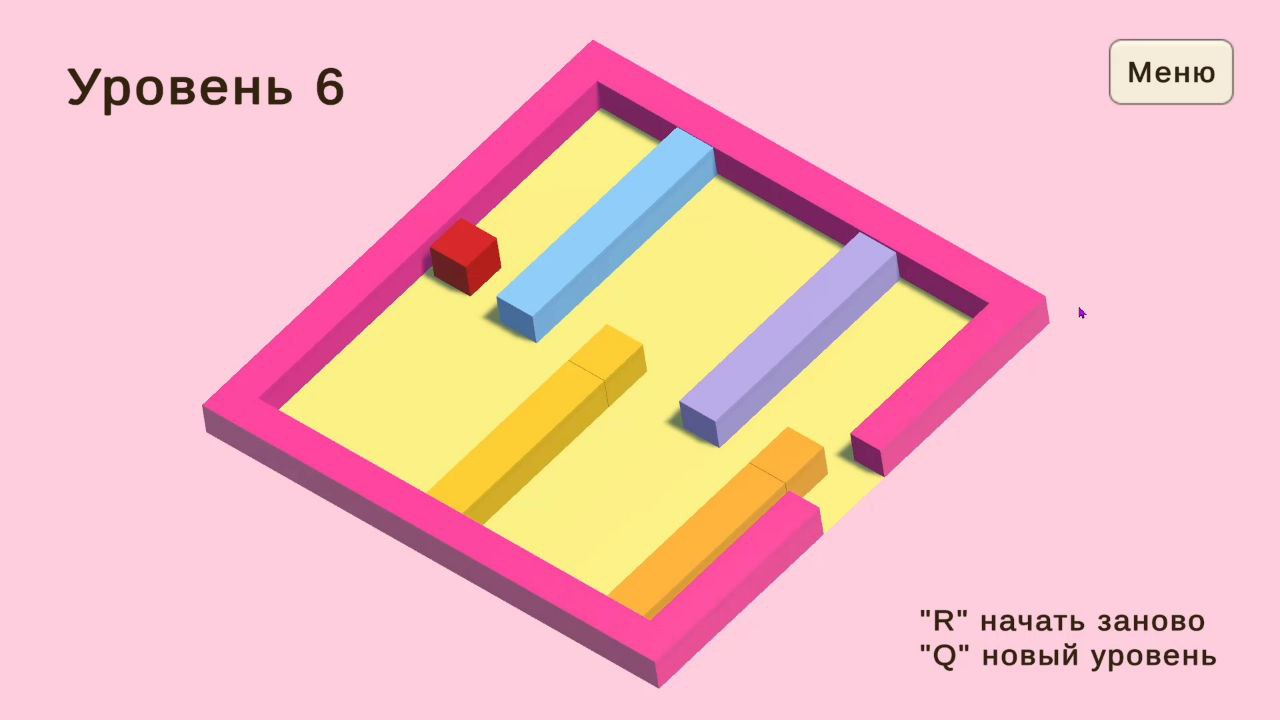


Рисунок 12 (Уровень 6)

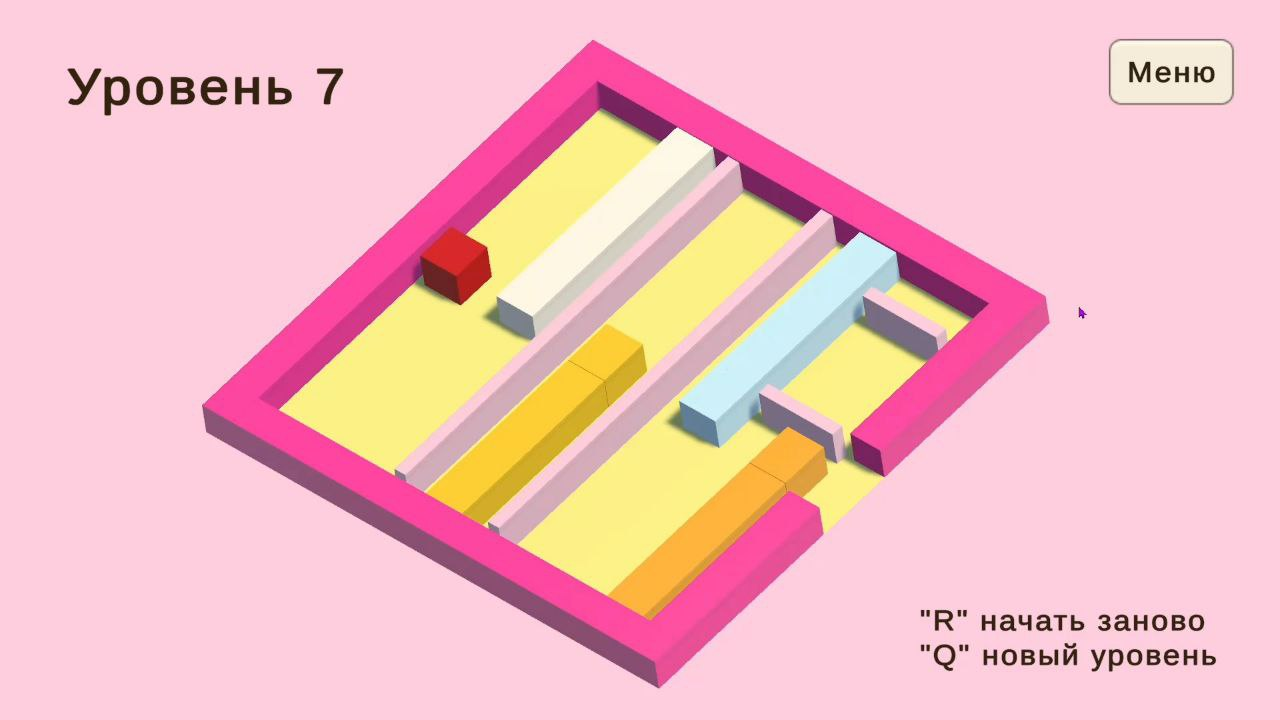


Рисунок 13 (Уровень 7)

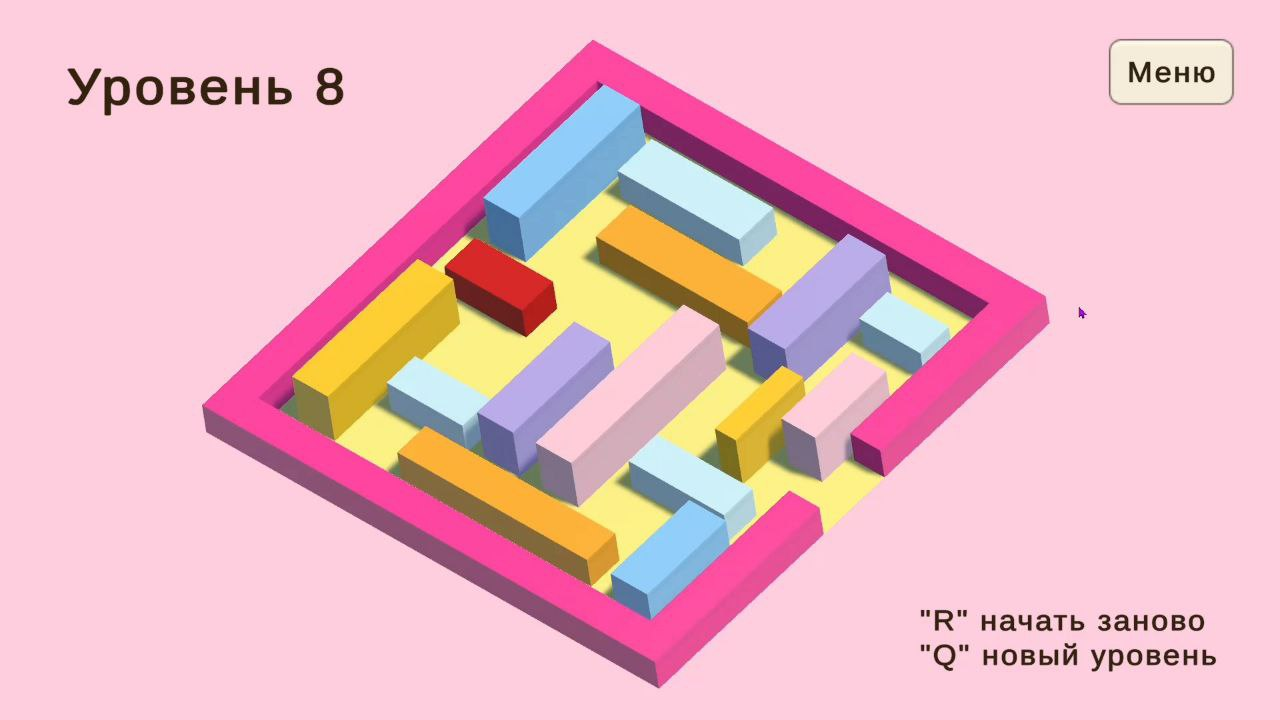


Рисунок 14 (Уровень 8)



Рисунок 15 (Выключение звука)



Рисунок 16 (Игра пройдена)

**Приложение 2.**Программный код 1(Player)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Player : MonoBehaviour

{

[SerializeField] KeyCode keyOne;

[SerializeField] KeyCode keyTwo;

[SerializeField] Vector3 moveDirection;

[SerializeField] AudioClip moveSound;

private AudioSource audioSource;

private void Start()

{

audioSource = gameObject.AddComponent<AudioSource>();

audioSource.playOnAwake = false;

// 💾 Сохраняем текущий уровень

PlayerPrefs.SetInt("lastLevel", SceneManager.GetActiveScene().buildIndex);

}

[System.Obsolete]

private void FixedUpdate()

{

bool moved = false;

if (Input.GetKey(keyOne))

{

GetComponent<Rigidbody>().velocity += moveDirection;

moved = true;

}

if (Input.GetKey(keyTwo))

{

GetComponent<Rigidbody>().velocity -= moveDirection;

moved = true;

}

// 🎵 Воспроизведение звука при движении

if (moved && !audioSource.isPlaying && moveSound != null)

{

audioSource.PlayOneShot(moveSound);

}

if (Input.GetKey(KeyCode.R))

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

if (Input.GetKey(KeyCode.Q))

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

}

[System.Obsolete]

private void OnTriggerEnter(Collider other)

{

if (this.CompareTag("Player") && other.CompareTag("Finish"))

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

if (this.CompareTag("Cube") && other.CompareTag("Cube"))

{

foreach(Activator button in FindObjectsOfType<Activator>())

{

button.canPush = false;

}

}

}

[System.Obsolete]

private void OnTriggerExit(Collider other)

{

if (this.CompareTag("Cube") && other.CompareTag("Cube"))

{

foreach (Activator button in FindObjectsOfType<Activator>())

{

button.canPush = true;

}

}

}

public void GoToMainMenu()

{

SceneManager.LoadScene("MainMenu"); // или индекс 0, если это первая сцена

}

}

Программный код 2(MainMenu)

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class MainMenu : MonoBehaviour

{

public Toggle muteToggle;

public AudioSource music;

private void Start()

{

// Загрузить предыдущее состояние звука

bool isMuted = PlayerPrefs.GetInt("Muted", 0) == 1;

AudioListener.pause = isMuted;

muteToggle.isOn = isMuted;

}

public void OnStartGame()

{

SceneManager.LoadScene("Level1"); // Или другая игровая сцена

}

public void OnQuitGame()

{

Application.Quit();

}

public void OnMuteToggleChanged()

{

bool mute = muteToggle.isOn;

AudioListener.pause = mute;

PlayerPrefs.SetInt("Muted", mute ? 1 : 0);

PlayerPrefs.Save();

}

public void ReturnToLastLevel()

{

int lastLevel = PlayerPrefs.GetInt("lastLevel", 1); // 1 — по умолчанию, если ничего не сохранено

SceneManager.LoadScene(lastLevel);

}

}

Программный код 3(Activator)

using Unity.VisualScripting;

using UnityEngine;

public class Activator : MonoBehaviour

{

public GameObject[] firstGroup;

public GameObject[] secondGroup;

public Activator button;

public Material normal;

public Material transparent;

public bool canPush;

private void OnTriggerEnter(Collider other)

{

if (canPush)

{

if (other.CompareTag("Cube") || other.CompareTag("Player"))

{

foreach (GameObject first in firstGroup)

{

first.GetComponent<Renderer>().material = normal;

first.GetComponent<Collider>().isTrigger = false;

}

foreach (GameObject second in secondGroup)

{

second.GetComponent<Renderer>().material = transparent;

second.GetComponent<Collider>().isTrigger = true;

}

GetComponent<Renderer>().material = transparent;

button.GetComponent<Renderer>().material = normal;

button.canPush = true;

}

}

}

}

**Приложение 3.**

Ссылка на презентацию: <https://docs.google.com/presentation/d/1epa49yLvoXwZEXqY-Z0TB15UnF9BHFgSaLDiXFxa94g/edit?usp=sharing>

**Приложение 4.**

Ссылка на репозиторий: <https://github.com/violetta-viola/BlockFlow>

**Приложение 5.**

Ссылка на видео-демонстрацию(презентация): <https://drive.google.com/drive/folders/1o1bUM0FuIjltlS9PgbKN5ZFGBByWkdw6?usp=sharing>

**Приложение 6.**

Ссылка на файл программы с ассетами и конфигурацией NEAT: <https://drive.google.com/drive/folders/1BviXsyn_GMXDesYTAMQCF9TYHHZM21JQ?usp=sharing>

**Приложение 7.**

Ссылка на видео-прохождение игры: <https://drive.google.com/drive/folders/1GLMjR_ekVqQ5FQRge1vvMBd0RtnvvHcD?usp=sharing>