Министерство образования Калининградской области

государственное автономное учреждение

Калининградской области

профессиональная образовательная организация

«Колледж предпринимательства»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ (ДИПЛОМНАЯ) РАБОТА**

**Тема: Игра**

|  |  |
| --- | --- |
| Выпускная квалификационная  (дипломная) работа  допущена к защите  Заместитель директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_ Бурыкина Ю.И.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. | Выполнил:  обучающийся группы ИСП19-21  специальность 09.02.02 Компьютерные сети  \_\_\_\_\_\_\_\_\_\_\_ Родин А.А.  Руководитель:  \_\_\_\_\_\_\_\_\_\_\_ Преподавателев А.А. |

Калининград

2023**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc134081843)

[**1.Теоретическая часть** 4](#_Toc134081844)

[**1.1** **объектно-ориентированный язык программирования общего назначения C#** 6](#_Toc134081845)

[**1.2 Navigation Mesh** 7](#_Toc134081846)

[**1.3** **Жанр Стратегии** 8](#_Toc134081847)

[**1.2.1 Классификация стратегий** 9](#_Toc134081848)

[**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 24](#_Toc134081849)

[**ПРИЛОЖЕНИЯ** 25](#_Toc134081850)

# **ВВЕДЕНИЕ**

В современном мире большую популярность набирает индустрия видео игр, при это самыми популярными направлениями являются многопользовательские, условно бесплатные (с возможностью внутри игровых транзакций) и с элементами коллекционирования виртуальных объектов проекты. Из-за чего сегмент однопользовательских проектов, ориентированных на долгое погружение в историю, изучение механик и бросающих вызов пользователю вымирает. Одними из таких частей стали стратегии – упрощение, попытка ввести внутри игровые транзакции, и концентрация на визуальной части привели к падению продаж, потери целевой аудитории у крупных издателей, и как результат серьёзное падение популярности жанра. Так же проекты данного жанра считаются экономически не выгодными, ввиду больших затрат для производства и малого спроса, так как целевая аудитория не доверяет крупным издателям и внимательно следит за публикуемой информацией о проектах. При этом спрос на стратегии не упал, что послужило хорошей почвой для маленьких команд разработчиков, что выпускают одну игру по шаблонам старых игр или же меняют привычные механики на совсем новые и необычные.

* Целью данной работы является создание видео игры в жанре Стратегии, в под жанре RTS, по шаблону старых RTS. Были поставлены следующие цели:
* создание базовых механик передвижения, управления и стрельбы у игровых пешек;
* создание игрового интерфейса;
* создание механик строительства, ремонта и захвата строений;
* реализация созданных механик и интерфейса в проект;

# **1.Теоретическая часть**

Для осуществления, запланированного необходимо выбрать “игровой движок” - базовое [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [компьютерной игры](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений.

Среди вариантов были:

**Unity** (*unity* в переводе с [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) — «единство», произносится как «ю́нити») — [кроссплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), разработанная американской компанией [Unity Technologies](https://ru.wikipedia.org/wiki/Unity_Technologies). Написан на С++, С#. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих [персональные компьютеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), [игровые консоли](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0), [мобильные устройства](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), [интернет-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и другие. Выпуск Unity состоялся в [2005](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80%D0%B0%D1%85) году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие [визуальной среды разработки](https://ru.wikipedia.org/wiki/RAD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [межплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних [библиотек](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и [независимыми студиями](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8-%D0%B8%D0%B3%D1%80%D0%B0). В итоге простой интерфейс и удобные функции, но подходит в основном для небольших игр.

**Unreal Engine** — [игровой движок](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA), разрабатываемый и поддерживаемый компанией [Epic Games](https://ru.wikipedia.org/wiki/Epic_Games). Написанный на языке [C++](https://ru.wikipedia.org/wiki/C%2B%2B), движок позволяет создавать игры для большинства [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и платформ: [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows), [Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux), [Mac OS](https://ru.wikipedia.org/wiki/Mac_OS" \o "Mac OS) и [Mac OS X](https://ru.wikipedia.org/wiki/Mac_OS_X" \o "Mac OS X); [консолей](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0) [Xbox](https://ru.wikipedia.org/wiki/Xbox" \o "Xbox), [Xbox 360](https://ru.wikipedia.org/wiki/Xbox_360" \o "Xbox 360), [Xbox One](https://ru.wikipedia.org/wiki/Xbox_One" \o "Xbox One), [PlayStation 2](https://ru.wikipedia.org/wiki/PlayStation_2" \o "PlayStation 2), [PlayStation 3](https://ru.wikipedia.org/wiki/PlayStation_3" \o "PlayStation 3), [PlayStation 4](https://ru.wikipedia.org/wiki/PlayStation_4), [PlayStation 5](https://ru.wikipedia.org/wiki/PlayStation_5" \o "PlayStation 5), [PSP](https://ru.wikipedia.org/wiki/PlayStation_Portable), [PS Vita](https://ru.wikipedia.org/wiki/PS_Vita), [Wii](https://ru.wikipedia.org/wiki/Wii" \o "Wii), [Dreamcast](https://ru.wikipedia.org/wiki/Dreamcast" \o "Dreamcast), [GameCube](https://ru.wikipedia.org/wiki/Nintendo_GameCube" \o "Nintendo GameCube) и др., а также на различных портативных устройствах, например, устройствах [Apple](https://ru.wikipedia.org/wiki/Apple" \o "Apple) ([iPad](https://ru.wikipedia.org/wiki/IPad" \o "IPad), [iPhone](https://ru.wikipedia.org/wiki/IPhone" \o "IPhone)), управляемых системой [iOS](https://ru.wikipedia.org/wiki/IOS" \o "IOS) и прочих. (Впервые работа с [iOS](https://ru.wikipedia.org/wiki/IOS" \o "IOS) была представлена в 2009 году, в 2010 году продемонстрирована работа движка на устройстве с системой [webOS](https://ru.wikipedia.org/wiki/Palm_webOS" \o "Palm webOS)).

Для упрощения портирования движок использует модульную систему зависимых компонентов; поддерживает различные системы [рендеринга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3) ([Direct3D](https://ru.wikipedia.org/wiki/Direct3D), [OpenGL](https://ru.wikipedia.org/wiki/OpenGL" \o "OpenGL), [Pixomatic](https://ru.wikipedia.org/w/index.php?title=Pixomatic&action=edit&redlink=1" \o "Pixomatic (страница отсутствует)); в ранних версиях: [Glide](https://ru.wikipedia.org/w/index.php?title=Glide_API&action=edit&redlink=1" \o "Glide API (страница отсутствует)), [S3](https://ru.wikipedia.org/w/index.php?title=S3_Metal&action=edit&redlink=1), [PowerVR](https://ru.wikipedia.org/w/index.php?title=PowerVR_SGL&action=edit&redlink=1" \o "PowerVR SGL (страница отсутствует))), воспроизведения звука ([EAX](https://ru.wikipedia.org/wiki/Environmental_Audio_Extensions), [OpenAL](https://ru.wikipedia.org/wiki/OpenAL" \o "), [DirectSound3D](https://ru.wikipedia.org/wiki/DirectSound3D); ранее: [A3D](https://ru.wikipedia.org/w/index.php?title=A3D&action=edit&redlink=1)), средства голосового воспроизведения текста, распознавание речи, модули для работы с сетью и поддержки различных устройств ввода.

Для игры по сети поддерживаются технологии [Windows Live](https://ru.wikipedia.org/wiki/Windows_Live" \o "Windows Live), [Xbox Live](https://ru.wikipedia.org/wiki/Xbox_Live), [GameSpy](https://ru.wikipedia.org/wiki/GameSpy" \o "GameSpy) и прочие, включая до 64 игроков (клиентов) одновременно. Таким образом, движок адаптировали и для применения в играх жанра [MMORPG](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BE%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D0%B8%D0%B3%D1%80%D0%B0). В итоге возможность реализации качественной графики, хорошая масштабируемость.

**Godot Engine** ([фр.](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BD%D1%86%D1%83%D0%B7%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *gó-doh* ) — [открытый](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [кроссплатформенный](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) 2D и 3D [игровой движок](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA) под [лицензией MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT), который разрабатывается сообществом Godot Engine Community.

Задача Godot — быть максимально интегрированной и самодостаточной средой для [разработки игр](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B2%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80). Среда позволяет разработчикам создавать игры с нуля, не пользуясь более никакими инструментами, за исключением тех, которые необходимы для создания игрового контента (элементы графики, музыкальные треки и т. д.). Процесс программирования также не требует внешних инструментов (хотя при необходимости использовать внешний редактор, это можно сделать относительно легко).

Общая архитектура движка построена вокруг концепции [дерева](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) из наследуемых «сцен». Каждый элемент сцены ([нода](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%88%D0%B8%D0%BD%D0%B0_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)" \o "Вершина (теория графов))), в любой момент сам может стать полноценной сценой. Поэтому при разработке можно легко изменять полностью всю архитектуру проекта, расширять её элементы в любую сторону и работать с комплексными сценами на уровне простых абстракций.

Все игровые ресурсы, от скриптов до графических ассетов и игровых сцен, хранятся в папке проекта как обычные файлы, и не являются частью сложной базы данных проекта. Ресурсы, которые не представляют собой комплексных данных, хранятся в простых текстовых форматах (например, скрипты и сцены, в отличие от моделей и текстур). Эти решения позволяют значительно упростить разным командам разработчиков работу с [системами управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). В итоге самодостаточен, бесплатен, но плохо подходит для создания 3д игр.

Исходя из поставленной задачи был выбран UNITY, ввиду простого интерфейса и удобных функций для разработчика. Исходя из этого игра будет написана на языке программирования C#.

# [**объектно-ориентированный**](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)[**язык программирования**](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)**общего назначения C#**

C# (произносится *си шарп*) — [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения. Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) под руководством [Андерса Хейлсберга](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) и Скотта Вильтаумота[[6]](https://ru.wikipedia.org/wiki/C_Sharp" \l "cite_note-6) как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework) и [.NET Core](https://ru.wikipedia.org/wiki/.NET_Core). Впоследствии был стандартизирован как [ECMA](https://ru.wikipedia.org/wiki/ECMA)-334 и [ISO](https://ru.wikipedia.org/wiki/ISO)/[IEC](https://ru.wikipedia.org/wiki/IEC) 23270.

C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java" \o "Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

Переняв многое от своих предшественников — языков [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Delphi (язык программирования)), [Модула](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) и, в особенности, [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает [множественное наследование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественная реализация интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%8)).

## **1.2 Navigation Mesh**

Навигационная сетка — это набор двумерных [выпуклых многоугольников](https://en.wikipedia.org/wiki/Convex_polygon) ( [полигональная сетка](https://en.wikipedia.org/wiki/Polygon_mesh) ), которые определяют, какие области среды могут проходить агенты. Другими словами, персонаж в игре может свободно ходить по этим областям, не встречая препятствий из-за деревьев, лавы или других барьеров, являющихся частью окружающей среды. Соседние полигоны соединяются друг с другом в [графе](https://en.wikipedia.org/wiki/Graph_data_structure) .

Поиск пути в пределах одного из этих многоугольников может быть тривиально выполнен по прямой линии, потому что многоугольник выпуклый и проходимый. Поиск пути между полигонами в сетке можно выполнить с помощью одного из большого количества алгоритмов [поиска по графу](https://en.wikipedia.org/wiki/Graph_search). Таким образом, агенты в навигационной сетке могут избежать дорогостоящих проверок [обнаружения столкновений](https://en.wikipedia.org/wiki/Collision_detection) с препятствиями, которые являются частью среды.

Представление проходимых областей в двумерной форме упрощает расчеты, которые в противном случае пришлось бы выполнять в «настоящей» трехмерной среде, но, в отличие от двумерной сетки, позволяет проходимым областям перекрываться сверху и снизу на разной высоте. Многоугольники различных размеров и форм в навигационных сетках могут отображать произвольные среды с большей точностью, чем обычные сетки.

## **Жанр Стратегии**

Стратегическая компьютерная игра ([разг.](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B8%D0%BB%D1%8C) «Стратегия») — один из основных жанров [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), в котором игроку для победы необходимо применять [стратегическое мышление](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F). В популярных играх такого жанра игроку часто предлагается играть не за конкретного персонажа, а за их условные массы, к примеру, руководить строительством города или командовать целыми армиями в военных кампаниях.

Хотя многие другие типы видеоигр могут содержать стратегические элементы, в собственно стратегических играх основное внимание уделяется планированию высшего уровня, логистике и управлению ресурсами. Стратегии обычно делятся на две основные подкатегории: [пошаговые](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F) и [в реальном времени](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_%D0%B2_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8), но есть также ряд кросс-/поджанров стратегий, которые содержат дополнительные элементы, такие как тактика, дипломатия, экономика и исследование.

К стратегическим играм относят также абстрактные (*[Ataxx](https://en.wikipedia.org/wiki/Ataxx" \o "en:Ataxx)*  (англ.)[рус.](https://ru.wikipedia.org/w/index.php?title=Ataxx&action=edit&redlink=1)), настольные ([шахматы](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D1%85%D0%BC%D0%B0%D1%82%D1%8B), [шашки](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D1%88%D0%BA%D0%B8), «[Монополия](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%BE%D0%BF%D0%BE%D0%BB%D0%B8%D1%8F_(%D0%B8%D0%B3%D1%80%D0%B0))»), с симуляцией менеджмента (“[*M.U.L.E.*](https://ru.wikipedia.org/wiki/M.U.L.E.)*”*, “*[Spaceward Ho!](https://en.wikipedia.org/wiki/Spaceward_Ho!" \o "en:Spaceward Ho!)”*  (англ.)[рус.](https://ru.wikipedia.org/w/index.php?title=Spaceward_Ho!&action=edit&redlink=1)) и другие; особенности некоторых из них были переняты разработчиками сюжетных компьютерных стратегий в реалистичном, историческом или фантастическом [сеттингах](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%82%D0%B8%D0%BD%D0%B3" \o "Сеттинг).

### **1.2.1 Классификация стратегий**

Все компьютерные стратегии относятся к двум основным типам:

* стратегии в реальном времени (RTS, от англ. Real Time Strategy)
* пошаговые стратегии (TBS, от англ. Turn Based Strategy).

Классификация по времени действия относится лишь к принципу временной организации в игре и не имеет никакого отношения к присутствию или отсутствию таких элементов геймплея, как строительство базы, добыча ресурсов и подобных.

#### **Пошаговые стратегии/TBS**

TBS от англ. Turn-Based Strategy. Пошаговые стратегии — игры, в которых игроки производят свои действия по очереди. Пошаговые стратегии появились раньше RTS и отличаются значительно большим разнообразием. Разделение игрового процесса на ходы отрывает его от реальной жизни и лишает игру динамизма, в результате чего эти игры не так популярны, как стратегии в реальном времени. С другой стороны, в TBS у игрока гораздо больше времени на размышление, во время совершения хода его ничто не торопит, что позволяет уделять больше времени планированию.

Особый подвид пошаговых стратегий представляют игра “Scorched Earth” и произошедшая от неё серия “Worms”. В них много общего с жанром аркады, логическое мышление и выбор правильных решений почти не требуется, игрок управляет очень небольшим числом персонажей (в “Worms” — командой из нескольких червяков) или вообще одним (“Scorched Earth”). Тем не менее, эти игры относят к пошаговым стратегиям.

#### **Стратегии в реальном времени/RTS**

RTS от англ. Real Time Strategy. В этих стратегиях игроки производят свои действия одновременно. Они появились несколько позже пошаговых, а первой получившей известность игрой этого жанра была “[Dune II](https://ru.wikipedia.org/wiki/Dune_II" \o "Dune II)” ([1992](https://ru.wikipedia.org/wiki/1992_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%B8%D0%B3%D1%80%D0%B0%D1%85)), сюжет которой основан на [одноимённом произведении](https://ru.wikipedia.org/wiki/%D0%94%D1%8E%D0%BD%D0%B0_(%D1%80%D0%BE%D0%BC%D0%B0%D0%BD)) [Фрэнка Герберта](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%80%D0%B1%D0%B5%D1%80%D1%82,_%D0%A4%D1%80%D1%8D%D0%BD%D0%BA). В настоящее время жанр стратегий в реальном времени стал весьма широк и включает игры совершенно разного типажа – от тактических варгеймов до глобальных стратегий, поэтому классические стратегии в реальном времени вроде “Dune” традиционно называют "классическими RTS". Особый подвид представляют собой стратегии в реальном времени с активной паузой, где игроку предоставляется возможность поставить игру на паузу и раздать необходимые приказы (например, “[Europa Universalis IV](https://ru.wikipedia.org/wiki/Europa_Universalis_IV" \o "Europa Universalis IV)”).

Во времена выхода “[Dune II](https://ru.wikipedia.org/wiki/Dune_II" \o "Dune II)” сформировались общие принципы классических стратегий в реальном времени (часть из них общие с другими стратегиями):

* Экономика в игре носит вспомогательный характер и нацелена на сбор ресурсов;
* В основе экономики лежат здания, которые можно строить и разрушать. Именно они строят юниты (см. ниже) и проводят исследования. Иногда строительство производят особые юниты, иногда (как в “Dune”) — другие здания. Некоторые строения могут атаковать противника сразу после окончания строительства, некоторые после специальных усовершенствований;
* Сбор ресурсов осуществляется особыми юнитами в специально предусмотренных для этого местах (в “Dune” это поля [спайса](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%94%D1%8E%D0%BD%D0%B0))), после чего они переносятся к специальному зданию на базе и только после этого поступают в распоряжение игрока;
* Ресурсы могут быть разных видов (например, золото, древесина, деньги, металл, уголь) и расходуются на строительство юнитов и зданий (также на это уходит время);
* Юнит (unit) — любая боевая единица (пехотинец, танк, самолёт, корабль), которая обычно может атаковать другие юниты и здания и уничтожать их. Юниты имеют параметры, главными среди которых являются «[очки здоровья](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%BA%D0%B8_%D0%B7%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%D1%8C%D1%8F)», наносимые повреждения, броня (уменьшающая наносимый юниту урон), скорость. Юнит может включать в себя и несколько солдат (танков, артиллерийских орудий и т. д.), но в этом случае командовать ими по отдельности нельзя. Некоторые юниты не могут атаковать и предназначены для сбора ресурсов, строительства зданий, транспортировки других юнитов;
* Все здания и юниты имеют радиус обзора, дальше которого «видеть» они не могут;
* Игрок может видеть, что происходит только на тех территориях, что попадают в радиус обзора его строений и юнитов. Те территории, на которых он ещё не был, закрашены чёрным. Те, на которых его войска уже были, но в данный момент не могут их видеть, покрыты так называемым «[туманом войны](https://ru.wikipedia.org/wiki/%D0%A2%D1%83%D0%BC%D0%B0%D0%BD_%D0%B2%D0%BE%D0%B9%D0%BD%D1%8B)» (“Fog of War”), который обычно позволяет видеть ландшафт этой местности и находящиеся там здания (такими, какими они были в последний момент, когда их видели юниты игрока), но не позволяет видеть передвижения противника и связанные с этим возможные изменения ландшафта.

#### **Тактические стратегии**

Тактические стратегии представляют собой симуляцию тактики как в реальном времени, так и в пошаговом режиме, тем самым подразделяясь на жанры [RTT](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_%D0%B2_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8) (Real-Time Tactics) и [TBT](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0) (Turn-Based Tactics). В отличие от классических RTS, в данных стратегиях игроку не приходится заниматься менеджментом базы или добычей ресурсов, тем самым данные игры сконцентрированы в большей степени на тактике ведения боевых действий в условиях заранее заданного количества войск и/или ресурсов. Возможность получения подкреплений и дополнительных ресурсов в таких играх либо отсутствует вовсе, либо на ней не делается основной акцент. Примеры: “[World in Conflict](https://ru.wikipedia.org/wiki/World_in_Conflict)”, “Achtung! Cthulhu Tactics”, серия “[Jagged Alliance](https://ru.wikipedia.org/wiki/Jagged_Alliance_(%D1%81%D0%B5%D1%80%D0%B8%D1%8F_%D0%B8%D0%B3%D1%80))”.

#### **Браузерные онлайн-стратегии**

Стратегии, сочетающие основные элементы классической [RTS](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_%D0%B2_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8) с возможностью коллективной игры с большим количеством игроков по сети [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Примерами браузерных онлайн-стратегий могут служить игры “[Травиан](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%B2%D0%B8%D0%B0%D0%BD" \o "Травиан)”, “[My Lands](https://ru.wikipedia.org/wiki/My_Lands" \o "My Lands)”, “[OGame](https://ru.wikipedia.org/wiki/OGame" \o "OGame)”, “[Сфера Судьбы](https://ru.wikipedia.org/wiki/%D0%A1%D1%84%D0%B5%D1%80%D0%B0_%D0%A1%D1%83%D0%B4%D1%8C%D0%B1%D1%8B)”, “[Война племён](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B9%D0%BD%D0%B0_%D0%BF%D0%BB%D0%B5%D0%BC%D1%91%D0%BD)”.

#### **Симуляторы строительства и управления**

Симуляторы строительства и управления – жанр компьютерных игр, стоящий на стыке симуляторов и стратегий. Они могут быть посвящены строительству города, парка развлечений и других объектов, при этом предлагая экономические испытания в противовес военным и тактическим. Условно их можно разделить на несколько поджанров.

##### **Градостроительные стратегии**

Градостроительные стратегии (англ. city-building games) посвящены строительству городов и поселений. В них игрок занимается строительством собственного города и решает связанные с этим задачи и испытания – например, строит жильё и борется с автомобильными пробками.

##### **Экономические стратегии**

Экономические стратегии (англ. business simulation games; tycoon) — стратегии, в которых игрок занимается созданием и управлением собственным бизнесом. В таких стратегиях обычно более проработана экономическая составляющая, на которой и делается основной акцент. Иногда в таких стратегиях присутствуют соперники (конкуренты), но конкуренция с ними, в основном, экономическая.

Экономические стратегии бывают однопользовательские, в которых рыночная среда полностью моделируется только запрограммированными алгоритмами, и многопользовательские, в которых конкурентное окружение и рыночная среда формируется не только алгоритмами, но и в результате взаимодействия множества игроков, что делает виртуальную экономику и её поведение более реалистичным и непредсказуемым по сравнению с однопользовательскими играми. Одним из наиболее известных классических однопользовательских экономических симуляторов является игра “Capitalism” Тревора Чена. Среди многопользовательских экономических симуляторов можно выделить серию экономических браузерных игр от издателя [Виртономика](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0" \o "Виртономика), в которой пользователи развивают свой виртуальный бизнес в конкуренции и кооперации со множеством игроков, строят города, создают политические партии и управляют виртуальными государствами.

Наиболее распространены симуляторы бизнеса («тайкуны», от часто встречающегося в их названиях слова tycoon — «магнат»). Первой игрой подобного рода была игра [Сида Мейера](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B9%D0%B5%D1%80,_%D0%A1%D0%B8%D0%B4) “Railroad Tycoon”. В симуляторах бизнеса играющий управляет как огромными промышленными компаниями (“[Capitalism II](https://ru.wikipedia.org/wiki/Capitalism_II" \o "Capitalism II)”, “[Industry Giant II](https://ru.wikipedia.org/wiki/Industry_Giant_II" \o "Industry Giant II)”), так и более мелкими предприятиями, такими, как закусочная или зоопарк (такие симуляторы носят менее серьёзный характер и часто предназначены не для того, чтобы игрок долго просчитывал ситуации и принимал выгодные решения, а просто для его развлечения; проиграть в таких играх нелегко, так как противников в явном виде нет). Очень распространены транспортные экономические симуляторы, такие, как “[Transport Tycoon](https://ru.wikipedia.org/wiki/Transport_Tycoon" \o "Transport Tycoon)”.

Среди других видов экономических стратегий можно отметить такие поджанры, как [градостроительные симуляторы](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80) (наиболее известна породившая этот жанр игра “[SimCity](https://ru.wikipedia.org/wiki/SimCity" \o "SimCity)”, а также серия градостроительных игр, действие которых происходит в древнем мире, — “[Caesar III](https://ru.wikipedia.org/wiki/Caesar_III" \o "Caesar III)”, “[Zeus: Master of Olympus](https://ru.wikipedia.org/wiki/Zeus:_Master_of_Olympus" \o "Zeus: Master of Olympus)”, и прочие; в последней серии присутствуют военные действия), спортивные менеджеры (в которых игрок руководит спортивным клубом, обычно футбольным, но не участвует в матчах непосредственно, а выполняет роль тренера и менеджера команды), симуляторы жизни (очень специфический жанр, происходящий от игры “[The Sims](https://ru.wikipedia.org/wiki/The_Sims" \o "The Sims)”).

В связи со всеобщим глобальным развитием сети Интернет набирают силу экономические онлайн игры, то есть игры представляющие собой некий мир со своими правилами и законами, в котором игроку предоставляется возможность заниматься активной коммерческой деятельностью: строить торговые и производственные предприятия и сооружения, заниматься разработкой природных ресурсов, вести активную финансовую и банковскую деятельность в рамках игрового пространства.

Развитие технологий дистанционного онлайн обучения дали заметный толчок к использованию экономических онлайн игр и симуляторов в образовательных процессах. Корпорации и бизнес-школы начали активно использовать обучающие [экономические игры](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B) такие как “CapSim” или “[Simformer](https://ru.wikipedia.org/w/index.php?title=Simformer&action=edit&redlink=1" \o "Simformer (страница отсутствует))” в обучении студентов и менеджеров.

#### **Варгеймы**

Стратегии, в которых отсутствует экономическая составляющая. Обычно это игры, в которых играющий управляет отрядом или армией во время боя. Примером из популярных игр могут служить некоторые [моды](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B)) для “[Sudden Strike](https://ru.wikipedia.org/wiki/Sudden_Strike" \o "Sudden Strike)” и “[Блицкрига](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%B8%D1%86%D0%BA%D1%80%D0%B8%D0%B3_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0))”.

«Варгейм» — не жанр, а характеристика конкретной игры, которая говорит — вам не придётся считать в ней деньги, большую часть игрового времени вы будете воевать всеми доступными средствами. А вот какие средства окажутся вам доступны, и что надо для этого сделать, зависит от фантазии разработчиков.

Примером такой игры является “[Z](https://ru.wikipedia.org/wiki/Z_(%D0%B8%D0%B3%D1%80%D0%B0))” [1996 года](https://ru.wikipedia.org/wiki/1996_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%B8%D0%B3%D1%80%D0%B0%D1%85) от английской компании “[The Bitmap Brothers](https://ru.wikipedia.org/wiki/The_Bitmap_Brothers" \o "The Bitmap Brothers)”. Несмотря на полное отсутствие экономики, игроки без проблем повышают качество и количество своих отрядов, умело распоряжаясь двумя важнейшими игровыми ресурсами — территорией и временем. Для стратегического планирования товарно-денежные отношения вовсе не обязательны, хотя, традиционны. Сам жанр остаётся всё той же стратегией или тактикой.

#### **Глобальные стратегии**

В глобальных стратегиях игрок принимает участие в целом периоде мировой истории, управляя государством; при этом игра учитывает различные факторы развития государств — экономический, военный, культурный, социальный и политический. Выделение глобальных стратегий в единый жанр характерно для русскоязычных специалистов; на [Западе](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B8%D1%80) этот жанр обычно разделяют на два — [4X](https://ru.wikipedia.org/wiki/4X) и [grand strategy](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%BE%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F" \l "%D0%A2%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F" \o "Глобальная стратегия). Глобальные стратегии могут быть как в реальном времени, так и пошаговые.

#### **4X стратегии**

Геймплей 4Х стратегий включает 4 полноценно реализованных элемента: исследование (новых земель), расширение, эксплуатация, уничтожение. Название происходит от буквы X в соответствующих данным понятиям английских словах eXplore, eXpand, eXploit, eXterminate. Классическим примером 4X стратегий является серия “[Civilization](https://ru.wikipedia.org/wiki/Civilization" \o "Civilization)”. В 4X стратегиях, также как и в глобальных, игрок зачастую управляет целым государством. Однако это не является обязательным критерием. 4X стратегии могут быть как в реальном времени, так и пошаговые.

##### Отличие 4X стратегий от глобальных стратегий

Отличие 4X стратегий от глобальных стратегий состоит в том, что в глобальных стратегиях может отсутствовать или быть слабо выраженным один из X-элементов. Например, глобальная стратегия “Hearts of Iron IV” не содержит в себе элемента eXplore (исследование новых земель). Это означает, что стратегия может быть одновременно 4X и глобальной, но не каждая глобальная стратегия является 4X.

#### **MOBA**

Игры жанра MOBA ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Multiplayer Online Battle Arena) сочетают в себе элементы [стратегий в реальном времени](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_%D0%B2_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8) и [компьютерных ролевых игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0). Этот жанр возник как ответвление (через [модификации](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B))) стратегий в реальном времени, и многие игры жанра MOBA используют схожий с ними интерфейс, но отличаются особыми правилами сражения.

В играх жанра MOBA две команды игроков сражаются друг с другом на карте особого вида. Каждый игрок управляет одним персонажем из определённого списка доступных героев, отличающихся способностями. В течение матча персонажи могут становиться сильнее, получать новые способности и снаряжение, подобно [компьютерным ролевым играм](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0). Конечной целью в ходе матча является уничтожение главного здания вражеской команды с помощью как героев, управляемых игроками, так и юнитов, управляемых компьютером.

#### **Карточные стратегии**

Игры, схожие с [настольными](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) [коллекционными карточными играми](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BA%D0%B0%D1%80%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0). Пример: “[Magic The Gathering](https://ru.wikipedia.org/wiki/Magic_The_Gathering" \o "Magic The Gathering).”

**2.1**

Ввиду того, что в UNITY обращение к скриптам в одном скрипте приходиться использовать имя скрипта из-за чего для некоторых юнитов были созданы дополнительные скрипты, являющиеся копией уже существующих. Например, скрипт обнаружения целей для танка и артиллерии одинаковы, но из-за того, что наведением на цель орудия у танка и артиллерии отвечает два разных скрипта и сделать один код который может сам обращаться к двум разным скриптам с разными наименованиями не получилось.

Передвижение юнитом происходит с помощью navMech. С помощью кода при нажатии левой кнопки мыши от положения курсора на камере создаётся луч сталкивающийся с игровыми объектами и проверяющий по заданным условиям, а именно – наличие третьего слоя в атрибутах и проверка тэга объекта, если тэг “unit” то у этого объекта болевое значение “selected”, отвечающее за выбор объекта равняется “true”. Если нажата правая кнопка мыши и объект выбран то положения курсора на камере создаётся луч и в точке контакта с каким либо объектом создаётся точка к которой объекту нужно двигатся.

Код в файле “Control Unit” (смотрите в приложении) отвечает за управление игровой техникой, а именно определить какую технику игрок выбрал и куда нажал на землю, чтобы создать на этом месте точку в которую техника будет передвигаться при помощи navmesh. Так же код сбрасывает статус выбранных у любой техники, если игрок нажал левой клавишей мышки на не технику.

Код в файлах “artTarget” и “tankTarget” одинаковы за исключением строчки с переменной t, это связано с тем, что в файлах “Turrel” и “TurrelArt” имели разные классы и по которым к ним можно обращаться. Сам код проверяет какие объекты вошли в прозрачную сферу вокруг техники и собирает в список целей все объекты в файле “Storega” параметр команда такого же параметра в технике, к которой файлы “artTarget” и “tankTarget” приклеплены.

Код файла “Build” создаёт сетку вокруг игрового здания, которая создаётся на основании полученных значений ширены и длины из параметра “Size”. Также два метода: “OnDrawGizmosSelected” и “SetNormal”, первый создаёт кубы, которые видны только разработчику, для отображение занимаемой зданием области, второй перекрашивает здание в исходный цвет, используются в файле “Building”.

using UnityEngine;

public class build : MonoBehaviour

{

public Vector2Int Size = Vector2Int.one;

public Renderer Mainrenderer;

public int costBuild;

public void SetTransparent(bool avaible)

{

if (avaible)

{

Mainrenderer.material.color = Color.green;

}

else

{

Mainrenderer.material.color = Color.red;

}

}

public void SetNormal()

{

Mainrenderer.material.color = Color.white;

}

private void OnDrawGizmosSelected()

{

// Отрисовка сетки здания

for(int x = 0; x < Size.x; x++)

{

for(int y = 0; y < Size.y; y++)

{

Gizmos.color = new Color(0f, 1f, 0f, 0.3f); //Цвет

Gizmos.DrawCube((transform.position) + new Vector3(x, 0, y), new Vector3(1,.1f,1)); //Отрисовка куба

}

}

}

}

Код файлы “Building” (смотрите в приложении) отвечает логику строительства, создание области где можно разместить строение, перемещение здания по полю на одну условную клетку, так как строение перемещается на 1000 единиц измерения в “Unity”, перемещение здания за курсором мышки, осуществляемое при помощи проведение луча от курсора в область куда направлена камера и когда луч касается земли в это место перемещается здание, проверка на возможность строительства, осуществляется проверкой массива “grid”, в котором записаны координаты зданий и сами здания, изменение цвета здания в зависимости от возможности установить его на землю, установка здания на область и запись координат и этого строения в массив.

Код файла “BulletF” отвечает за нанесение урона игровым объектам, осуществляемое обращением к файлу “Storega" и уменьшения параметра “hp” после чего пуля удаляется из игры.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class bulletF : MonoBehaviour

{

public int damage;

private void OnTriggerEnter(Collider other)

{

if (other != null && other.gameObject.layer == 3)

{

if(other.gameObject.tag != "earth")

{

other.gameObject.GetComponentInParent<storega>().hp -= damage;

}

Destroy(gameObject);

}

}

}

Код файла “factoryScript” (смотрите в приложении) открывает кнопки для создания игровой техники при условии, если игрок выбрал завод. При получении образца техники, создаёт её в точке “pointSpawn”, только если у игрока хватает денег.

Код файла “SpawnBut” привязан к кнопкам создания техники, при их нажатии происходит поиск всех выбранных заводов и передаёт им образец техники для создания.

using System.Collections;

using System.Collections.Generic;

using Unity.VisualScripting;

using UnityEngine;

public class spawnBut : MonoBehaviour

{

public void SpawnBut(GameObject unit)

{

var arr = FindObjectsOfType(typeof(factoryScript));

for (int i = 0; i < arr.Length; i++)

{

if (arr[i].GetComponent<factoryScript>().selected)

{

arr[i].GetComponent<factoryScript>().test = unit;

}

}

}

}

Код файла “playerScript” перемещает и вращает объект, если игрок нажимает клавиши “ц”, “ф”, “ы”, “в”, “й”, “у”, “к”, “а” на который смотрит камера по игровой сцене и показывает количества игровой валюты.

Файла “Storega" хранит переменные, и если здоровье объекта меньше или равно нулю.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class storega : MonoBehaviour

{

public int cost;

public int hp;

public string team;

// Update is called once per frame

void Update()

{

if(hp <= 0)

{

Destroy(gameObject);

}

}

}

Код файла “TurrelArt” (смотрите в приложении) отвечает за наведение башни и орудия, создание снаряда и придание ему такой скорости, с которой снаряд долетит до цели. Расчёт происходит по формуле v= √(gx2)/(2(y-tgα\*x) \* cos2α, где v – скорость, g – ускорение свободно падения, у и x это дистанция до цели на осях y и x соответственно, α – это угол под которым снаряд вылетает из ствола.

Код файла “Turrel” (смотрите в приложении) отвечает за наведение башни и орудия на цель осуществляется получением направления на цель, вычислением насколько нужно развернуть башню и орудие, чтобы они смотрели на цель, после чего создаётся снаряд постоянной скоростью. Поворот башни в файле “TurrelArt” осуществляется точно также.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. [Программирование на C# для начинающих](https://www.litres.ru/aleksey-nikolaevich-/programmirovanie-na-c-dlya-nachinauschih-osn-34338191/?lfrom=973672683)
2. [Head First. Изучаем C#](https://www.litres.ru/dzhennifer-grin-2/izuchaem-c-8479459/?lfrom=973672683)
3. [Unity и C#. Геймдев от идеи до реализации](https://www.litres.ru/dzheremi-gibson-bond/unity-i-s-geymdev-ot-idei-do-realizacii-pdf-64085991/?lfrom=973672683)
4. https://ru.wikipedia.org/wiki/C\_Sharp
5. https://ru.wikipedia.org/wiki/Unreal\_Engine
6. https://ru.wikipedia.org/wiki/Godot
7. https://ru.wikipedia.org/wiki/Игровой\_движок#cite\_note-\_e5e20b91561a71fd-2
8. <https://ru.wikipedia.org/wiki/Unity_(игровой_движок)>
9. <https://en.wikipedia.org/wiki/Navigation_mesh>
10. [https://ru.wikipedia.org/wiki/Компьютерная\_стратегическая\_игра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)

# **ПРИЛОЖЕНИЯ**

ПРИЛОЖЕНИЕ А

Код в файле “Control Unit”

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEditor.PackageManager;

using UnityEngine;

using UnityEngine.AI;

using static UnityEngine.UI.Image;

public class ControlUnit : MonoBehaviour

{

public NavMeshAgent agent;

public bool select = false;

public Transform enemy;

public Vector3 navtarger;

//public List<GameObject> selectGroup = new List<GameObject> ();

void Start()

{

}

// Update is called once per frame

void Update()

{

// Выбор юнита

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ А

if (Input.GetMouseButton(0) && Input.GetKey(KeyCode.LeftShift))// Проверка на выбор группы техники при помощи комбинации: Левый шифт + левая кнопка мыши

{

Debug.Log("OWWO");

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

if (Physics.Raycast(movePosition, out var hitInfo, Mathf.Infinity, 1 << 3))

{

if (hitInfo.collider.gameObject.transform.parent.CompareTag("unit") && hitInfo.collider.gameObject.GetComponentInParent<storega>().team == "player")

{

hitInfo.rigidbody.gameObject.GetComponentInParent<ControlUnit>().select = true;

}

else

{

select = false;

}

}

}

else if (Input.GetMouseButtonDown(0))

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ А

{

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

if(Physics.Raycast(movePosition, out var hitInfo, Mathf.Infinity, 1<<3))

{

if (hitInfo.collider.gameObject.CompareTag("earth")) { }

else if (hitInfo.collider.gameObject.transform.parent.CompareTag("unit") && hitInfo.collider.gameObject.GetComponentInParent<storega>().team == "player")

{

select = false;

hitInfo.rigidbody.gameObject.GetComponentInParent<ControlUnit>().select = true;

}else{

select = false;

}

}

}

if (Input.GetMouseButtonDown(1) && select == true)

{

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ А

if(Physics.Raycast(movePosition, out var hitInfo))

{

SetPointNav(hitInfo.point);

}

}

}

public void SetPointNav(Vector3 pos)

{

navtarger = pos;

agent.SetDestination(pos);

}

}

ПРИЛОЖЕНИЕ Б

Код файлы “Building”

public class building : MonoBehaviour

{

public Vector2Int GridSize = new Vector2Int(400, 400);

private build[,] grid;

private build flyingBuilding;

private Camera mainCamera;

public playerScript player;

private void Awake()

{

grid = new build[GridSize.x, GridSize.y];

mainCamera = Camera.main;

player = FindObjectOfType<playerScript>();

}

public void StartPlacingBuilding(build buildPref)

{

if(flyingBuilding != null)

{

Destroy(flyingBuilding.gameObject);

}

flyingBuilding = Instantiate(buildPref);

flyingBuilding.gameObject.GetComponent<NavMeshObstacle>().enabled = false;

}

private void Update()

{

if (flyingBuilding != null)

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ Б

{

var groundPlane = new Plane(Vector3.up, Vector3.zero);

Ray ray = mainCamera.ScreenPointToRay(Input.mousePosition);

if(groundPlane.Raycast(ray, out float position))

{

Vector3 worldPosition = ray.GetPoint(position);

int x = Mathf.RoundToInt(worldPosition.x/100)\*100;

int y = Mathf.RoundToInt(worldPosition.z/100)\*100;

bool available = true;

if(x < 0 || x > GridSize.x - flyingBuilding.Size.x) available = false;

if(y < 0 || y > GridSize.x - flyingBuilding.Size.y) available = false;

if(available && IsPlaceTaker(x, y)) available = false;

flyingBuilding.transform.position = new Vector3(x, 0, y);

flyingBuilding.SetTransparent(available);

if (available && Input.GetMouseButtonDown(0))

{

PlaceFlyingBuilding(x, y);

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ Б

}

}

}

}

private bool IsPlaceTaker(int placeX, int placeY)

{

for (int i = 0; i < flyingBuilding.Size.x; i++)

{

for (int y = 0; y < flyingBuilding.Size.y; y++)

{

if(grid[placeX + placeX, placeY] != null)

{

return true;

}

}

}

return false;

}

private void PlaceFlyingBuilding(int placeX, int placeY)

{

if (flyingBuilding.gameObject.GetComponent<build>().costBuild <= player.cash)

ПРИЛОЖЕНИЕ Б

ПРОДОЛЖЕНИЕ

{ player.cash -= flyingBuilding.gameObject.GetComponent<build>().costBuild;

for (int i = 0; i < flyingBuilding.Size.x; i++)

{

for (int y = 0; y < flyingBuilding.Size.y; y++)

{

grid[placeX + placeX, placeY] = flyingBuilding;

}

}

flyingBuilding.SetNormal();

flyingBuilding.gameObject.GetComponent<NavMeshObstacle>().enabled = true; flyingBuilding.gameObject.GetComponent<storega>().team = "player";

flyingBuilding = null;

}

}

}

ПРИЛОЖЕНИЕ В

Код файла “factoryScript”

public class factoryScript : MonoBehaviour

{

public GameObject pointSpawn;

public Canvas canvas;

public GameObject test;

public bool selected = false;

public bool clickInterfase;

public GraphicRaycaster raycaster;

PointerEventData m\_PointerEventData;

EventSystem m\_EventSystem;

private GameObject b;

public playerScript player;

void Start()

{

canvas = FindObjectOfType<Canvas>();

test = null;

raycaster = canvas.GetComponent<GraphicRaycaster>();

m\_EventSystem = GetComponent<EventSystem>();

player = FindObjectOfType<playerScript>();

}

void Update()

{

if (Input.GetMouseButtonDown(0))

{

clickInterfase = false;

m\_PointerEventData = new PointerEventData(m\_EventSystem);

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ В

m\_PointerEventData.position = Input.mousePosition;

List<RaycastResult> results = new List<RaycastResult>();

графического raycast по положению щелчка

raycaster.Raycast(m\_PointerEventData, results);

foreach (RaycastResult result in results)

{

clickInterfase = true;

}

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

if (Physics.Raycast(movePosition, out var hitInfo, Mathf.Infinity, (1 << 3 ) | (1 << 5)))

{

GameObject q = hitInfo.collider.gameObject;

if (q.CompareTag("factory") || clickInterfase)

{

q.GetComponent<factoryScript>().selected = true;

}

else

{

selected = false;

}

UnityEngine.Debug.Log(q);

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ В

for (int i = 2; i < canvas.transform.childCount; i++)

{

if (q.CompareTag("factory") || clickInterfase)

{

if (canvas.transform.GetChild(i).name != "factoryMenu")

{ canvas.transform.GetChild(i).gameObject.SetActive(false);

}

else

{ canvas.transform.GetChild(i).gameObject.SetActive(true);

}

}

else

{

if (canvas.transform.GetChild(i).name != "factoryMenu")

{

canvas.transform.GetChild(i).gameObject.SetActive(true);

}

Else

{

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ В

canvas.transform.GetChild(i).gameObject.SetActive(false);

}

}

}

}

}

if (selected && test != null && player.cash >= test.GetComponent<storega>().cost)

{

player.cash -= test.GetComponent<storega>().cost;

b = Instantiate(test, pointSpawn.transform.position, pointSpawn.transform.rotation);

b.GetComponent<storega>().team = gameObject.GetComponent<storega>().team;

test = null;

}

}

}

ПРИЛОЖЕНИЕ Г

Код файла “TurrelArt”

public class TurrelArt : MonoBehaviour

{

public Transform shootEl;

public Transform navBulletPos;

public Transform LookAtObj;

public float dmg = 10;

public float shootSpeed;

public GameObject bullet;

public Transform target;

public bool isShoot;

public GameObject GunTank;

public float SpeedRotation = 90;

public Transform PointRotateGun;

public float AngleInDegrees;

public float g = Physics.gravity.y;

void Update()

{

if(target != null)

{

Vector3 navAtTargetXZ = Vector2d("y");

Quaternion targetRotation = Quaternion.LookRotation(navAtTargetXZ); Квартерионы

transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, SpeedRotation \* Time.deltaTime);

PointRotateGun.localEulerAngles = new Vector3(-AngleInDegrees, 0f, 0f);

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ Г

if (isShoot == false)

{

StartCoroutine(Shoot());

}

}

}

}

public Vector3 Vector2d(string axis)

{

Vector3 nav = target.position - transform.position; //Получение направления на цель

switch (axis)

{

case "x":

nav.x = 0;

break;

case "y":

nav.y = 0;

break;

case "z":

nav.z = 0;

break;

}

return nav;

}

IEnumerator Shoot()

{

isShoot = true;

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ Г

yield return new WaitForSeconds(shootSpeed);

Vector3 nav = target.position - shootEl.position;

Vector3 navXZ = new Vector3(nav.x, 0f, nav.z);

float x = navXZ.magnitude;

float y = nav.y;

float AngleInRadians = AngleInDegrees \* Mathf.PI / 180;

float v2 = (g \* x \* x) / (2 \* (y - Mathf.Tan(AngleInRadians) \* x) \* Mathf.Pow(Mathf.Cos(AngleInRadians), 2));

float v = Mathf.Sqrt(Mathf.Abs(v2));

GameObject b = Instantiate(bullet, shootEl.position, shootEl.rotation);

b.GetComponent<Rigidbody>().velocity = shootEl.transform.forward \* v;

isShoot = false;

}

}

ПРИЛОЖЕНИЕ Д

Код файла “Turrel”

public class Turrel : MonoBehaviour

{

public Transform shootEl;

public Transform navBulletPos;

public Transform LookAtObj;

public float dmg = 10;

public float shootSpeed;

public GameObject bullet;

public Transform target;

public bool isShoot;

public GameObject GunTank;

public float SpeedRotation = 90;

public Transform PointRotateGun;

public float AngleInDegrees;

void Update()

{

if(target != null)

{

Vector3 navAtTargetXZ = Vector2d("y");

Quaternion targetRotation = Quaternion.LookRotation(navAtTargetXZ, Vector3.up); Квартерионы

transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, SpeedRotation \* Time.deltaTime);

// PointRotateGun.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, SpeedRotation \* Time.deltaTime);

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ Д

Vector3 navAtTargetYZ = target.position - transform.position;

Quaternion t = Quaternion.LookRotation(navAtTargetYZ);

PointRotateGun.rotation = Quaternion.RotateTowards(PointRotateGun.rotation, t, 10 \* Time.deltaTime);

Ray traicShell = new Ray(shootEl.position, shootEl.forward);

Debug.DrawRay(shootEl.position, shootEl.forward);

RaycastHit hit;

if (Physics.Raycast(traicShell, out hit, Mathf.Infinity, (1 << 3)))

{

if (hit.collider.gameObject.transform.parent.gameObject == target.gameObject)

{

if (isShoot == false)

{

StartCoroutine(Shoot());

}

}

}

}

}

public Vector3 Vector2d(string axis)

{

ПРОДОЛЖЕНИЕ

ПРИЛОЖЕНИЕ Д

Vector3 nav = target.position - transform.position;

switch (axis)

{

case "x":

nav.x = 0;

break;

case "y":

nav.y = 0;

break;

case "z":

nav.z = 0;

break;

}

return nav;

}

IEnumerator Shoot()

{

isShoot = true;

yield return new WaitForSeconds(shootSpeed);

Vector3 n = navBulletPos.position - shootEl.position;

GameObject b = GameObject.Instantiate(bullet, shootEl.position, transform.rotation) as GameObject;

b.GetComponent<Rigidbody>().velocity = n \* 1;

isShoot = false;

Debug.Log("qwe");

}

}