Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б07-мм

Создание 3D-игры, сочетающей жанры «tower defense» и «tower offence»

Хечнев Семен Евгеньевич

Отчёт по учебной практике в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к. т. н., Ю. В. Литвинов

Санкт-Петербург
2023

Оглавление

Введение

Видеоигры в настоящее время играют большую роль в жизни человека. Согласно данным DFC Intelligence[1], в видеоигры играет 3.1 млрд человек или около 40% населения Земли. Из них 1.5 млрд играют в игры, используя компьютер. Ожидается, что к 2026 году мировая игровая индустрия будет стоить 321 миллиард долларов, согласно отчету PwC Global Entertainment and Media Outlook 2022-26[2]. Исходя из этих данных можно сделать вывод о том, что разработка собственной компьютерной видеоигры является перспективным направлением.

Жанр «tower defense» является поджанром стратегических игр. В классической игре данного жанра цель игрока (защищающегося) — не дать противникам (нападающим) дойти по заранее заданному пути до финиша, устанавливая вдоль пути башни, атакующие нападающих. Каждый уничтоженный враг даёт игроку ресурсы, обычно монеты; используя их, игрок может улучшить свои башни: увеличить урон, дальность атаки или построить новые башни для увеличения своей оборонительной мощи. Также в играх подобного жанра существуют волны врагов — с каждой волной враги становятся сильнее, возрастает их количество. Поэтому игрок должен каждый раз наращивать свой потенциал. Игрок побеждает после успешного прохождения всех волн. Проигрывает — в случае успешного достижения врагом финиша.

Кроме того, существует жанр «tower offence» — это противоположность «tower defense», где игрок управляет армией нападающих и пытается прорваться сквозь защиту.

В результате учебной практики планируется разработать прототип игры, сочетающий в себе жанры «tower defense» и «tower offence». Игрок будет выбирать за кого играть — за нападающего или защищающегося. Геймплей защищающегося не будет отличаться от классического. Нападающий будет управлять армией шаров. Цель нападающего — до-

[1] https://www.dfcint.com/global-video-game-consumer-population/ (дата доступа: 5 апреля 2023 г.).

[2] https://www.pwc.com/gx/en/industries/tmt/media/outlook/outlook-perspectives.html/ (дата доступа: 5 апреля 2023 г.).

вести армию шаров до финиша. Нападающий может атаковать башни защищающегося, взрывая шары вблизи башен. У нападающего будет два способа заработка монет: во-первых, пассивно, то есть каждую секунду будет начисляться некоторое количество монет, во-вторых, за уничтожение башни нападающий будет получать некоторое количество монет. Используя монеты, нападающий может улучшить свою армию: увеличить скорость передвижения, количество здоровья, размер наносимого урона по башням.

Игра будет отличаться простотой, относительно других стратегий, что привлекает большую аудиторию игроков, любящих не тратить много времени на изучение всех тонкостей и механизмов игры. Отсутствие жестоких сцен в игре позволит играть людям независимо от пола и возраста. Более того, игра будет сочетать в себе элементы стратегий и головоломок, поэтому она будет способствовать развитию логического мышления, что делает игровой процесс не только весёлым и интересным, но и полезным.

Дата сборки: 14 апреля 2023 г.

1. Постановка задачи

1.1. Цель

Целью работы является создание прототипа игры, включающей в себя характерные особенности жанров «tower defense» и «tower offence». Для её выполнения были поставлены следующие задачи.

1.2. Задачи осеннего семестра

1. Сделать обзор игровых движков, аналогичных игр.
2. Спроектировать базовую игровую логику: волны нападающих, экономика, здоровье замка (базы защищающегося).
3. Реализовать интерфейс защищающегося.
4. Реализовать основные механики игры за защищающегося: постройка, перемещение, улучшение башен, выбор приоритета атаки врага.
5. Разработать план апробации.

1.3. Задачи весеннего семестра

1. Реализовать основные механики игры за нападающего: выбор атакующих, участвующих в текущей волне; улучшение армии; возможность наносить при взрыве урон всем башням в некотором радиусе; выделение области и взрыв всех нападающих, находящихся в этой области; возможность пополнять здоровье нападающих в выбранной игроком области.
2. Добавить различные виды башен и атакующих.
3. Добавить звуки, анимации.
4. Провести апробацию.

2. Обзор

2.1. Существующие аналоги

Для обзора отбирались игры, которые относятся к жанру «tower defense» или «tower offence». Цель обзора — определение интересных механик, которые используются в играх подобного жанра.

2.1.1. Bloons TD 6

Bloons TD 6[3] — это одна из самых популярных игр в жанре «tower defense». Геймплей прост и не требует особых навыков для начала игры. В игре есть карта с маршрутом, по которому будут двигаться воздушные шары и дирижабли, задача игрока — выстроить защитные башни так, чтобы шары не достигли конца пути. Выстраивать защиту можно с помощью башен-обезьян и построек, атакующих шарики. В игре множество башен, а именно двадцать две, причём все башни уникальны. Среди них есть башни, которые наносят урон по области, замедляют шары, или не атакуют шары вовсе, а приносят игроку монеты.

Из этой игры была взята возможность строить башни, приносящие защищающемуся игроку монеты, башни, которые наносят урон сразу нескольким нападающим, и идея представления врагов в виде шариков.

2.1.2. Anomaly: Warzone Earth

Anomaly: Warzone Earth[4] — это одна из немногих игр жанра «tower offence». В игре предстоит управлять отрядом боевой техники, которую необходимо провести между инопланетными башнями, атакующими технику, до конечной точки. Техника самостоятельно выбирает целевую башню и атакует её. За уничтожение башни игрок получает деньги, на которые он может улучшить технику или пополнить отряд новыми боевыми единицами. Игрок может задавать маршрут, созда-

[3] https://store.steampowered.com/app/960090/Bloons_TD_6/ (дата доступа: 5 апреля 2023 г.).

[4] https://store.steampowered.com/app/91200/Anomaly_Warzone_Earth/ (дата доступа: 6 апреля 2023 г.).

вать дымовые завесы — области, в которых башни не могут атаковать технику игрока, создавать области регенерации, в которых техника восстанавливает здоровье.

Из данной игры была взята возможность атакующему игроку создавать регенерирующую область.

2.2. Существующие игровые движки

В первую очередь для обзора отбирались движки, которые удовлетворяют следующим критериям:

- программирование игровой логики на языке C# (так как автор знаком с C♯ и не хотел тратить время на изучение нового языка программирования);
- возможность создания кроссплатформенных игр;
- наличие бесплатной лицензии.

Среди популярных движков, подходящих под эти критерии, можно выделить Unity [4] и Godot [3].

Движки сравнивались по следующим критериями:

- количество учебных материалов;
- размер торговой площадки ассетов.

2.2.1. Unity

Unity [4] — игровой движок для разработки кроссплатформенных 2D или 3D игр. Разработан компанией Unity Technologies. Для программирования игровой логики используется язык C♯, так же определять игровую логику можно без написания кода при помощи системы визуального скриптинга[5]. В интернете можно найти множество туториалов для

[5] https://unity.com/ru/features/unity-visual-scripting (дата доступа: 9 апреля 2023 г.).

освоения движка, существует даже специальная платформа для обучения, поддерживаемая разработчиками движка — Unity Learn[6]. Движок бесплатен при условии, что доход не превышает 100 000 долларов в год. Чтобы не тратить время на создание собственных ассетов — различных наборов игровых ресурсов (3D/2D модели, звуки/музыка, шейдеры/частицы, наборы картинок, различные инструменты), есть официальная торговая платформа — Unity Asset Store[7], в которой можно найти более 84000 ассетов. Более того, для упрощения создания многопользовательских игр, сбора аналитики, привлечения новых игроков, внедрения монетизации существует официальная платформа игровых сервисов — Unity Gaming Services[8].

2.2.2. Godot

Godot [3] — движок для разработки кроссплатформенных 3D или 2D игр, разрабатываемый сообществом Godot Engine Community. Для программирования игровой логики используется собственный высокоуровневый, динамически типизированный язык GDScript, так же с версии 4.0 есть поддержка C♯ 8.0. Godot относительно недавно получил высокую популярность, поэтому обучающих материалов по движку заметно меньше, чем у Unity, но их всё равно достаточно для быстрого освоения. Движок полностью бесплатен, исходники движка выложены под лицензией MIT на GitHub[9]. Godot компактный — 74МБ. У движка есть официальная площадка ассетов — Godot Asset Library[10]. На площадке представлено более 1700 ассетов, при этом большинство из них под лицензий MIT.

[6] https://learn.unity.com/ (дата доступа: 9 апреля 2023 г.).
[7] https://assetstore.unity.com/ (дата доступа: 9 апреля 2023 г.).
[8] https://unity.com/solutions/gaming-services (дата доступа: 9 апреля 2023 г.).
[9] https://github.com/godotengine/godot (дата доступа: 9 апреля 2023 г.).
[10] https://godotengine.org/asset-library/asset (дата доступа: 9 апреля 2023 г.).

2.3. Использованные технологии

В качестве игрового движка был выбран Unity, потому что у движка большее сообщество, больше обучающего контента, магазин ассетов содержит больше контента.

Игра в Unity состоит из множества сцен — отдельных файлов, которые содержат в себе множество игровых объектов, а объекты в свою очередь содержат компоненты. Разработчик наполняет сцену игровыми объектами, а объекты — различными компонентами, которые определяют отображение и поведение объекта на сцене. Для того чтобы связать скрипт — файл с кодом, написанным на C♯, с игровым объектом, то есть сделать скрипт компонентом игрового объекта, необходимо создать класс, который наследуется от MonoBehavior. MonoBehavior — это базовый класс в Unity, который предоставляет методы жизненного цикла игрового объекта[11]. После создания файла с расширением «.cs», определения класса, определения поведения игрового объекта в методах жизненного цикла, необходимо выбрать игровой объект, который необходимо связать со скриптом, нажать в инспекторе — специальном окне редактора, предназначенном для настройки компонентов игрового объекта, на кнопку «Add Component» и выбрать из списка созданный файл. Так же для хранения информации независимо от экземпляров объектов на сцене можно создать класс, который наследуется от ScriptableObject, затем либо программно, либо в окне редактора «Project» можно создать экземпляр этого типа и использовать его как хранилище данных.

Согласно документации Unity [1]: «If you attempt to define a constructor for a script component, it will interfere with the normal operation of Unity and can cause major problems with the project». Поэтому актуальным становится использование фреймворка для внедрения зависимостей. Для внедрения зависимостей в Unity используют несколько фреймворков: StrangeIOC, VContainer, Extenject. Для разработки был использован Extenject[12], так как он до сих пор поддерживается, име-

[11] https://docs.unity3d.com/Manual/ExecutionOrder.html (дата доступа: 9 апреля 2023 г.).

[12] https://github.com/Mathijs-Bakker/Extenject (дата доступа: 6 апреля 2023 г.).

ет всеобъемлющую документацию с наглядными примерами, большую популярность среди аналогов и легко импортируется через Unity Asset Store.

Для создания базового уровня был использован ProBuilder — простой инструмент для быстрого моделирования карт.

3. Реализация

В первом семестре работа велась в основном над реализацией игровых механик и интерфейса защищающегося и базовой игровой логики.

3.1. Что было сделано

- Реализована базовая логика нападающих: здоровье, движение по маршруту.
- В игру добавлены две башни: обычная, которая атакует шарики, и приносящая игроку монеты.
- Реализовано строительство, перемещение башен. Каждая башня имеет уникальные атрибуты, каждый из них можно улучшить. Так же есть возможность выбрать приоритет атаки шариков по следующим критериями: близость к замку, количество здоровья.
- Реализована экономика. За постройку, перемещение, улучшение башни списывается определённое количество монет, за уничтожение шарика начисляется награда.
- Реализовано здоровье базы защищающегося — замка. Когда шарик касается замка, он наносит урон замку и уничтожается.
- Реализован базовый интерфейс защищающегося.
  - Кнопка для начала очередной волны.
  - Кнопки для постройки башен определённого типа.
  - Отображение количества монет.
  - Отображение количества здоровья у замка.
  - Панель, которая появляется при нажатии на башню, содержащая название башни, атрибуты, кнопки для улучшения определенного атрибута, закрытия панели, перемещения башни и выбора приоритета атаки шариков.

Рис. 1: Скриншот из игры

3.2. Архитектура

Для создания архитектуры использовался объектно-компонентный подход. Это значит, что поведение игрового объекта задаётся множеством привязанных к нему компонентов. Такой подход позволяет уменьшить связность кода и переиспользовать компоненты.

3.2.1. Башни

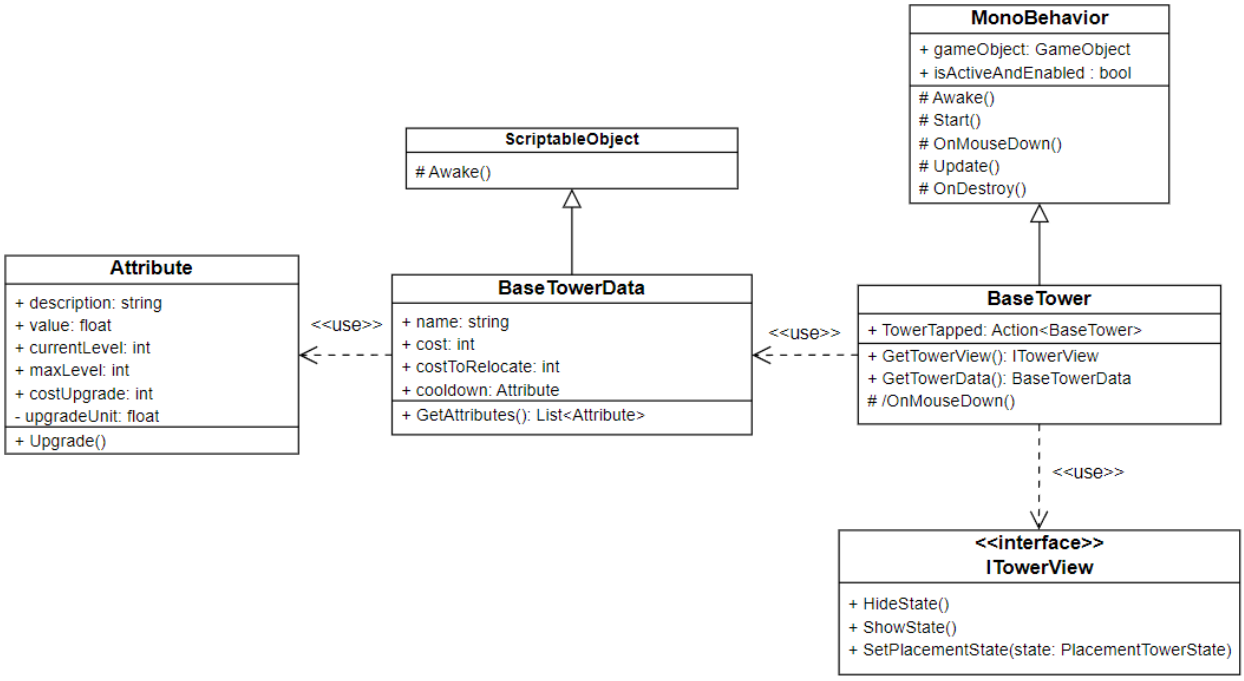


Рис. 2: Классы, определяющие поведение башен

Класс Attribute представляет из себя улучшаемый атрибут башни. Класс BaseTowerData содержит базовую информацию о башне: имя, стоимость постройки, стоимость перемещения, атрибут перезарядки, и абстрактный метод, который возвращает все атрибуты башни. Интерфейс ITowerView задаёт интерфейс взаимодействия с представлением башни. BaseTower является «контроллером» для башни и содержит ссылки на модель и представление и событие, вызываемое при нажатии на башню.

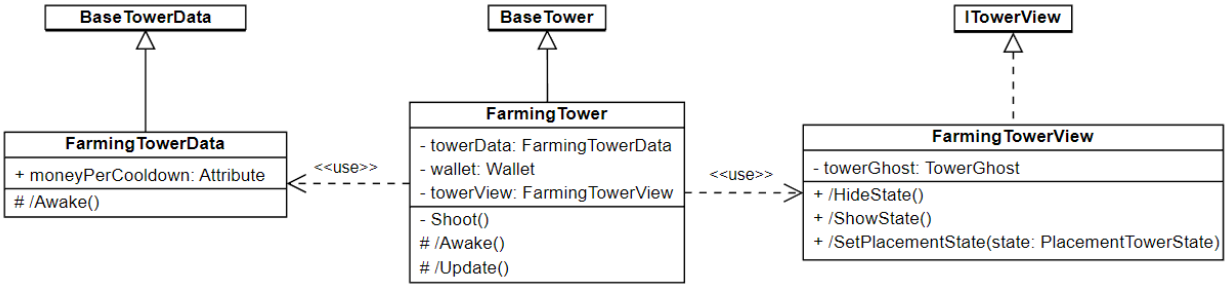


Рис. 3: Реализация башни, приносящей монеты

Атрибут moneyPerCooldown в классе FarmingTowerData определя-

ет количество монет, которые игрок будет получать каждый раз после перезарядки башни. FarmingTower добавляет раз в определенное количество секунд, равное значению атрибута cooldown, в wallet — кошелёк игрока, определенное количество монет, равное значению атрибута moneyPerCooldown. FarmingTowerView для отображения состояния использует компонент типа TowerGhost, который подсказывает игроку куда можно или нельзя поставить башню при постройке, меняя цвет игрового объекта башни на сцене на красный, который обозначает запрет постройки башни, или зелёный, обозначающий возможность установки башни.

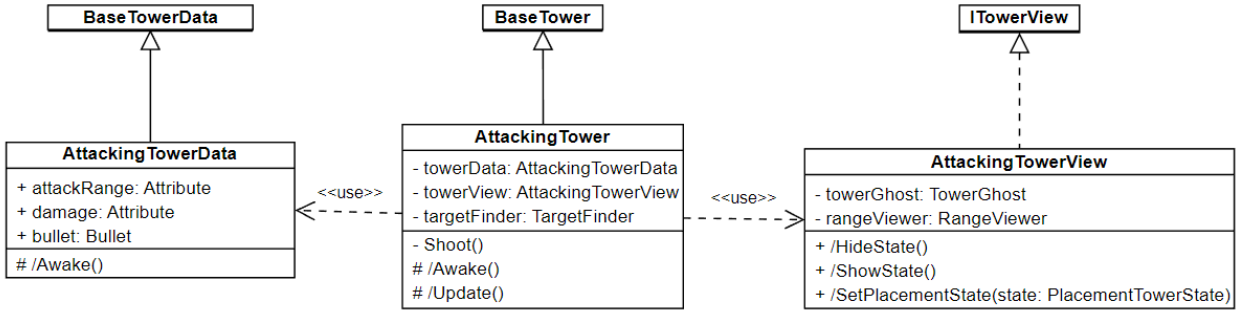


Рис. 4: Реализация башни, атакующей шарики

Атрибуты attackRange и damage в классе AttackingTowerData определяют дальность атаки и урон башни. AttackingTower получает цель при помощи targetFinder и запускает в неё снаряд — bullet, определенный в AttackingTowerData. AttackingTowerView кроме TowerGhost использует RangeViewer — компонент, рисующий окружность вокруг башни радиуса, равного значению attackRange.

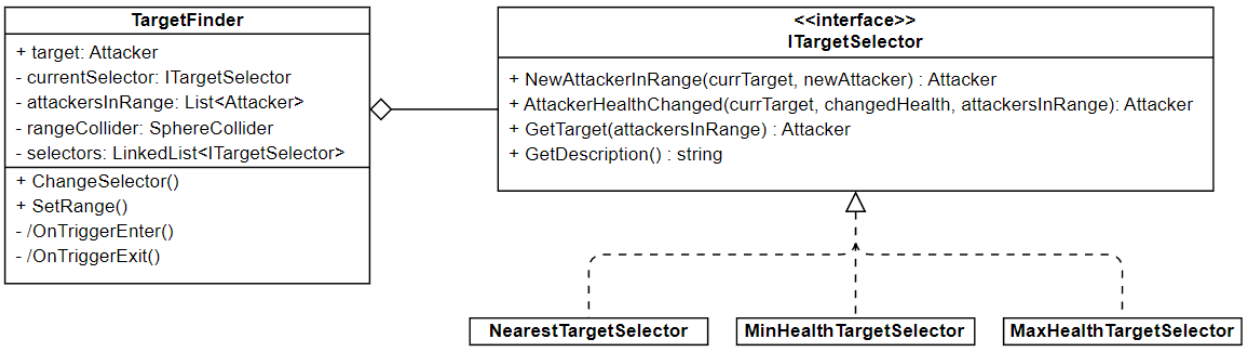
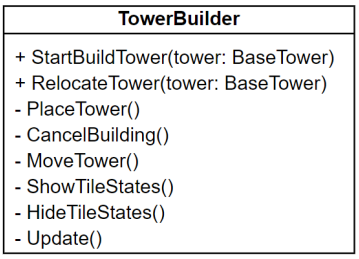


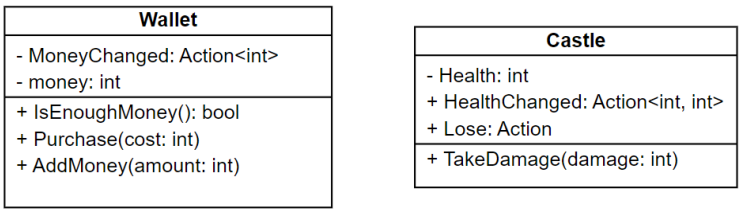
Рис. 5: Реализация компонента TargetFinder

TargetFinder предоставляет башне цель. Он отслеживает шарики, которые находятся в радиусе атаки башни, и делегирует выбор конкретной цели селектору — классу, который реализует интерфейс ITargetSelector. Реализовано три селектора: NearestTargetSelector выбирает ближайшую к замку цель, MinHealthTargetSelector — с минимальным количеством здоровья и MaxHealthTargetSelector — с максимальным.



TowerBuilder отвечает за постройку и перемещение башен.

3.2.2. Модели



- Wallet представляет модель кошелька игрока и реализует все операции с игровыми монетами.
- Castle представляет модель замка (базы защищающегося).

4. Апробация

Апробация запланирована на весенний семестр. План апробации: предложить друзьям поиграть в игру и оценить её по методике System Usability Scale [2].

Заключение

В итоге были достигнуты следующие результаты:

- выполнен обзор движков для создания игр, аналогичных игр;
- спроектирована базовая игровая логика;
- реализованы основные механики игры за защищающегося;
- реализован пользовательский интерфейс защищающегося;
- разработан план апробации.

Исходный код — https://github.com/s-khechnev/AttackAndDefend.

Список литературы

[1] Documentation Unity. Creating and Using Scripts. — URL: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html (дата обращения: 10 апреля 2023 г.).

[2] Wikipedia contributors. System usability scale — Wikipedia, The Free Encyclopedia. — 2023. — [Online; accessed 10-April-2023]. URL: https://en.wikipedia.org/w/index.php?title=System_usability_scale&oldid=1147509546.

[3] Википедия. Godot — Википедия, свободная энциклопедия. — 2023. — [Онлайн; загружено 10 апреля 2023]. URL: https://ru.wikipedia.org/?curid=7069975&oldid=129699107.

[4] Википедия. Unity (игровой движок) — Википедия, свободная энциклопедия. — 2023. — [Онлайн; загружено 05 апреля 2023]. URL: https://ru.wikipedia.org/?curid=1858166&oldid=129194130.