# Дипломна работа

на тема

RogueLike Tower Defense Игра

**……………………………………………………**

Дипломант: Константин Атанасов

ученик от 12А клас в

ЧПГДН "СофтУни Светлина"

Научен ръководител: **…………………………………..…………...**

Рецензент: …………………………………..……………

# Съдържание

[Дипломна работа 1](#_heading=h.30j0zll)

[Съдържание 2](#_heading=h.1fob9te)

[Увод 2](#_heading=h.3znysh7)

[Проблем 2](#_heading=h.2et92p0)

[Цели на дипломния проект 3](#_heading=h.tyjcwt)

[Глава 1. Проучване 3](#_heading=h.1t3h5sf)

[Глава 2. Проектиране и имплементация 5](#_heading=h.2s8eyo1)

[Използвани технологии 5](#_heading=h.e5y7ymvgclk)

[Unity 5](#_heading=h.77gg38z0itcz)

[C# 6](#_heading=h.v5gmy7vefxa4)

[Singleton pattern 6](#_heading=h.gaul7bjijbxj)

[Имплементация на системата 6](#_heading=h.3rdcrjn)

[Основни философии 6](#_heading=h.iey9osoukxw7)

[Основни скриптове и класове 10](#_heading=h.o8jepdcny99u)

[Managers 10](#_heading=h.2976mlfl2fqk)

[Ui Скриптове 11](#_heading=h.5c4lbfi0o5u7)

[Item скриптове 12](#_heading=h.woyyd2js8g98)

[Противници 13](#_heading=h.2o1j8dd6k36j)

[Тестване 15](#_heading=h.nz0de6w970c4)

[Внедряване (deployment) 15](#_heading=h.lnxbz9)

[Глава 3. Ръководство за потребителя 16](#_heading=h.35nkun2)

[Играене на играта 18](#_heading=h.gi96iq3zfeuv)

[Обяснение на основния екран на играта{Фиг 15}: 18](#_heading=h.2k0sd4h8bm03)

[Заключение 1](#_heading=h.44sinio)

[Информационни източници 1](#_heading=h.2jxsxqh)

[Рецензия на дипломен проект 1](#_heading=h.3j2qqm3)

# 

# Увод

Ръстът на услугите за развлечения като цяло и в по-голяма степен на видеоигрите показва, че хората се интересуват от неща, които ги забавляват. Това ме накара да избера видеоиграта за тема на дипломния ми проект. По-конкретно избрах комбинация от два жанра видеоигри: "RogueLike" и "Tower Defense". Първият, тъй като е доказано, че е популярен жанр чрез игри като "Binding of Isaac" и "Hades", а вторият, защото напоследък лично на мен ми харесва и затова исках да се предизвикам да комбинирам двете. За да създам проекта, реших да работя с Unity - водещ в индустрията енджин за игри.

## Проблем

Проблемът, който този проект решава, е просто скуката и нуждата от релаксация. В голямата схема на нещата това може да изглежда тривиално, но особено в днешния динамичен свят ние повече от всякога се нуждаем от начин да се отпуснем и да релаксираме след тежък работен ден. Това е допринесло за нарастващата популярност на видео игрите като цяло, което показва, че има постоянно нарастваща нужда от повече от тях

## Цели на дипломния проект

Тъй като моят проект е създаването на игра, следва, че той има много от същите цели като нормалната игра. А именно:

да бъде забавна

да отпуска играча

да бъде преиграваема

Освен това, когато започнах проекта, си поставих цели на проекта, които според мен ще бъдат това, което е необходимо, за да се нарече този проект завършен. Те включват:

Врагове, които вървят по пътя от началото до края

Различни видове врагове.

Възможност да се строят кули

Различни видове кули

Подобрения за кулите

Използвана система за инвентар, в която има всички кули и подобрения, които притежавате.

Екран, използван за надграждане на кулите.

Враговете пускат кули и подобрения.

Падналите оръжия и подобрения отиват в базата вместо в инвентара, което ви позволява да ги вземете или продадете.

Печелене на пари както при убиване на врагове, така и при продажба на кули

Смяна на нивата на всеки няколко вълни.

# Глава 1. Проучване

Тъй като основната цел на проекта е да съчетае двата жанра, първо разгледах всеки жанр поотделно.

Tower Defense:

Основните игри, които разгледах, бяха сериите "BloonsTD" и "Plants vs Zombies", тъй като те са най-популярните. Основните общи неща между тях са, че започвате на статична карта, която не се променя до края на нивото, и че имате различни кули и врагове. Разликата тук е, че в " Plants vs Zombies" местата на кулите са на пътя на враговете, докато в "Bloons" не можете да поставяте кули на пътя. Реших да избера метода на "Bloons", тъй като изграждам игра, която е по-близка до него, и следователно имаше повече смисъл.

RogueLike:

За да изуча този жанр, се обърнах към игрите "Binding of Isaac" и "Hades". И двете технически са класифицирани като RogueLite игри. Въпреки това двата жанра, както подсказват имената, са изключително близки. Всъщност единствената разлика е, че в игрите RogueLite можете да отключвате нови способности, предмети и да повишавате старта между игрите, докато в игрите RogueLike започвате с всичко отключено и всяка игра започва по един и същи начин. И двете игри споделят някои общи качества, като например факта, че разполагате с различни оръжия и че се биете в стаи, свързани по различни начини при всяко пускане. Разликата би била в това, че в "Binding of Isaac" можете да се връщате обратно през стаите, които вече сте изчистили, докато в "Hades" не можете.

Ето защо, за да създам RogueLike аспекти на моята игра, реших да направя всяко ниво като метафорична стая, в която оставате, докато не победите определен брой вълни от врагове, и след това напускате към следващата стая. За да имитирам различните оръжия, вместо това получавате различни кули, което се свързва с TowerDefense страната на играта.

Моята игра

И накрая, комбинирането на всичко това създава игра, която поддържа ниво с път през него като играта TowerDefense, но което се сменя с друго от време на време като играта RogueLike.

Използвате кули като в TowerDefense, но те се изпускат от враговете като в RogueLike.

Можете да придобивате различни подобрения като в "BloonsTD" и във всички RogueLike игри, които разгледах.

И накрая, както в RogueLike, не получавате никакви бонуси между игрите и всяка игра започва по един и същи начин.

# Глава 2. Проектиране и имплементация

## Използвани технологии

### Unity

Гейм енджинът Unity е популярен междуплатформен гейм енджин, който се използва за разработване и внедряване на видеоигри на различни платформи, като компютри, мобилни устройства, конзоли и устройства за виртуална реалност (VR). Той е пуснат за първи път през 2005 г. от Unity Technologies.

Unity също така има голяма общност от разработчици и потребители, която предоставя богати ресурси, уроци и поддръжка за разработчиците. Освен това Unity разполага със стабилен магазин за активи, където разработчиците могат да купуват активи, инструменти и плъгини, за да подобрят своите игри.

Това означаваше, че ако някога се затрудня и не знам, има по-голяма вероятност да намеря решение онлайн, отколкото ако използвах по-малко популярен двигател за игри. Всички тези неща заедно ме накараха да избера Unity като гейм енджин за този проект.

### C#

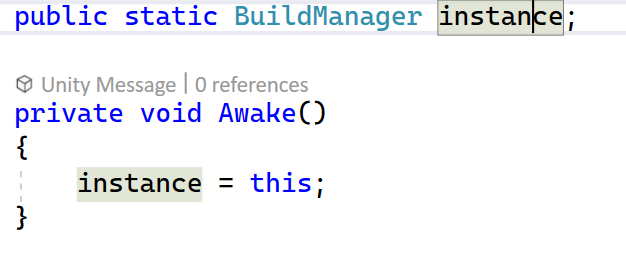
C# е обектно-ориентиран език за програмиране, разработен от Microsoft и пуснат за първи път през 2001 г. Той се използва широко за много неща, като например уеб приложения, настолни приложения и игри. C# е основният език за програмиране, с който работи unity. Той е и този, с който съм най-добре запознат, което ме накара да го избера за използване в този проект.

### Singleton pattern

Шаблонът Singleton е шаблон за проектиране, често използван при разработката на софтуер, който ограничава инстанцирането на даден клас до една инстанция и осигурява глобален достъп до тази инстанция. В Unity Singleton моделът може да се използва, за да се гарантира, че само една инстанция на определен скрипт съществува в играта във всеки един момент.

Използвам Singleton за различни управляващи скриптове или за скрипта на играча. Начинът, по който прилагам това, е чрез създаване на public static {className} instance променлива, която задавам в метода Awake на Unity на this. Както е показано в по-долу фигурата *Фиг1.*

Това означава, че ако някога ми се наложи да имам достъп до методи от този скрипт, мога да го направя, без да получавам отделна референция.

*фиг!.*

## Имплементация на системата

## Основни философии

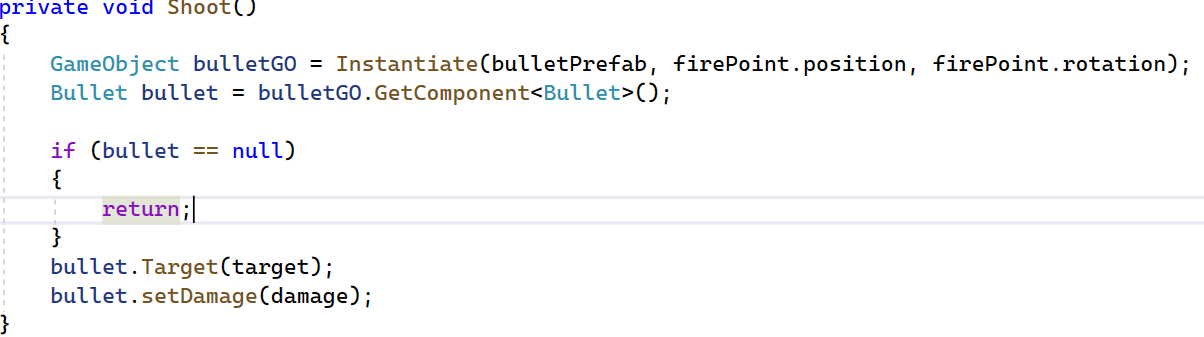
Това са основните философии, които се стремях да спазвам, докато пишех целия код на този проект. Целта им е да се създаде код, който е лесен за изграждане и с който се работи лесно.

1. **Четим код**:

Чрез именуване на всички функции, класове и променливи по начин, който описва тяхното предназначение. Това означава, че когато програмистът трябва да промени нещо, той може да знае точно какво прави всяка част от кода, което улеснява извършването на промени и

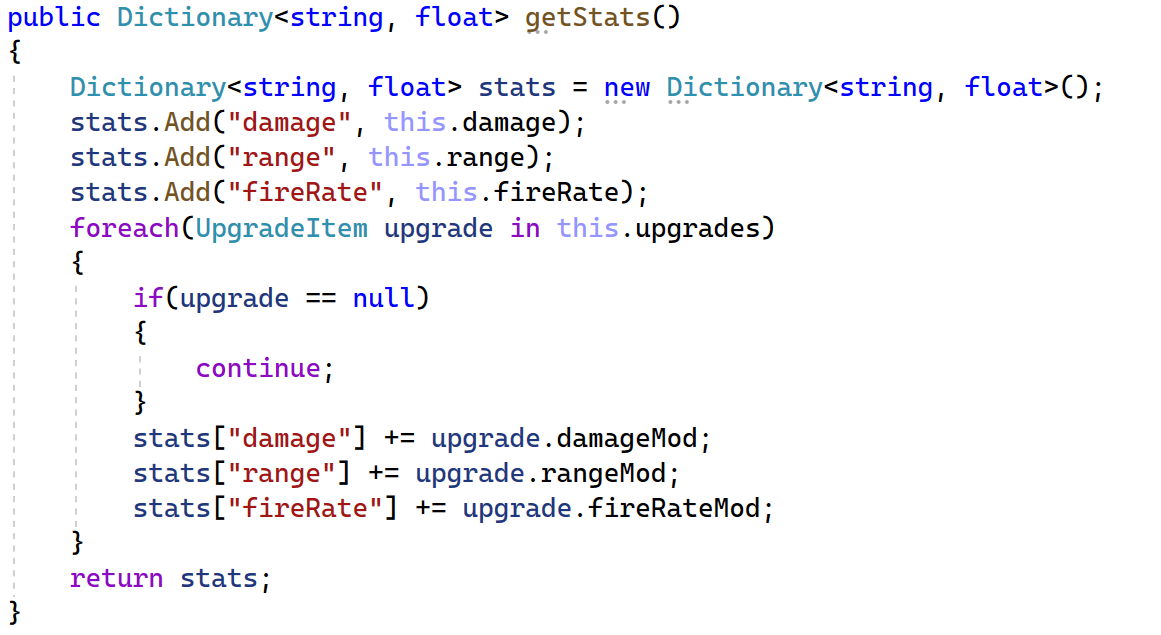
ли намирането на частта, която причинява проблеми. Това също така намалява необходимостта от коментари в кода. Следвайки тази философия, един добър код трябва да се нуждае от възможно най-малко коментари и въпреки това да бъде напълно четим и разбираем.

Снимка 2 е пример за това. Това е методът на стрелба за кули. Името му ви подсказва какво точно прави, а именно стрелба. След това всеки ред от кода се обяснява сам, без да са необходими коментари. Първо се инстанцира bulletPrefab. След това получава от него само скрипта Bullet. След това проверява дали той не е null, за да предотврати всякакви грешки. И накрая задава целта и щетите на куршума. Нито една част от това не е необходимо да се коментира и дори ако някой незапознат с моя проект прочете този код, той ще знае какво прави. Единственото изключение може да е, ако не е запознат с Unity и как работят Prefabs и Instantiating.



Фиг 2

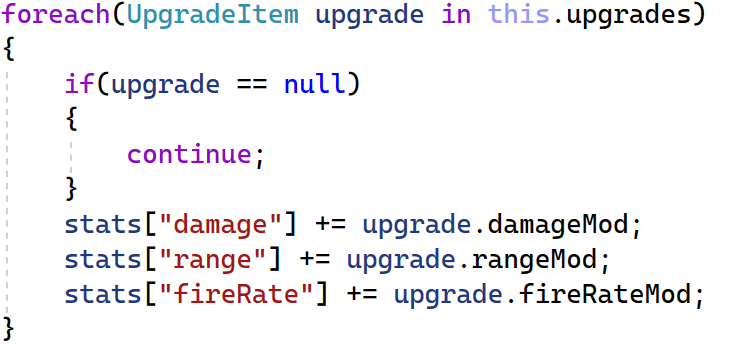
2. **Намаляване на нестване**:

В програмирането нестване се отнася до практиката на поставяне на един блок код или оператор в друг блок код или оператор. Ето един пример за код с 3 нива на нестване   
{*Фиг 3*} *****Фиг 3*

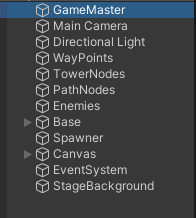
Засега кодът остава четим, затова 3 или най-много 4 нива на вмъкване са приемливи. След това кодът става нечетим, особено ако се използват if и else оператори, които могат да затруднят още повече проследяването на кода. Затова тази философия казва, че ако сте на толкова много нива, вероятно правите нещо погрешно, така че за да подобрите четимостта, можете да изнесете част от вътрешния код в отделна функция. Ако следвате предишната философия и наречете функцията така, че да е ясно какво прави, можете значително да подобрите четимостта на кода си

3. **използване на Guard Clauses (предпазни клаузи) и Early Return (ранно връщане)**:

това е практика на поставяне на if с неблагоприятното условие в началото на функцията и след това, ако това условие е изпълнено, функцията се прекратява рано, вместо да се продължи с останалата част от кода. Можете да видите пример за това в {*Фиг 4*}. this code could instead be written by putting the bottom three lines in the if and reversing the statement to (upgrade != null). This would however reduce the readability especially if we add more if statements in it later down the line. Making it more difficult to figure out where a certain input would lead to. While as the code is now you can easily tell that if the upgrade valuable is null the code will stop there, therefore everything below that if will only run if we don't enter the if statement. Additionally this also ties into the previous philosophy of reducing nesting since this pulls all of that code out of the if.

*Фиг 4*

4. **Използване на празни игрови обекти в Unity**: макар и да не е толкова свързано с кода, това също е нещо, което подобрява четимостта и използваемостта на проекта. "GameMaster" (ръководител на играта), "Enemies" (врагове), "PathNodes" (пътни възли), "TowerNodes" (кули), "Waypoints" (пътни точки), намерени в *Фиг 5*, са празни игрови обекти, които не можете да видите в играта. Всички те обаче служат за важни цели. Обектът "GameMaster" съдържа всички общи скриптове, необходими за стартирането на играта, като BuildManager, LevelSetup и WaveSpawner и други подобни. Останалите служат за различни цели. В началото те нямат деца, но винаги когато създавам враг например, след това го поставям като дете на празния обект "Enemies". Това не променя нищо визуално по време на игра, но когато го тествам в Unity, то прави интерфейса по-малко претрупан и улеснява навигацията.  
Въпреки че не е показано, слоят на потребителския интерфейс в Canvas съдържа и празни игрови елементи, които се използват за сортиране на други игрови елементи, така че е лесно да се разбере какво прави.

*Фиг 5*

# Основни скриптове и класове

Играта се състои от 2 основни сцени: MainMenu и LevelScene. Сцената MainMenu съдържа бутоните за стартиране на играта и за излизане от нея. След това бутонът за стартиране ще ви изпрати към сцената LevelScene, която съдържа по-голямата част от играта.

### Managers

Започнете с мениджърите в папката "Managers". Класът "BuildManager" съдържа цялата функционалност, свързана с изграждането на кули. Най-важна е променливата SelectedTurret, която пази в себе си коя кула планирате да построите. Заедно с нея има 2 функции GetSelectedTurret(), която връща TowerItem на кулата, и ChangeTurret(), която се използва за задаване на SelectedTurret. Последните 2 метода RemoveTurret и Deselect се използват, когато кулата е построена, за да 1. я премахнат от инвентара на играчите и 2. да зададат SelectedTurret на null, за да не можете да построите дубликати на въпросната кула.

Следва скриптът LevelSetup, който се използва за избор на произволно ниво и след това за изграждането му. всяко ниво е представено от 2 матрици и 2 вектора. Заедно те указват на скрипта LevelSetup къде да постави всеки възел и кои възли са за врагове и кои за кули. Всяко ниво се запазва като статично в класа Level.

Скриптовете за Player и Game са доста прости. Играчът съхранява информация за играча, като например колко пари има играчът, колко здраве има и какъв е неговият инвентар. Докато скриптът на играта контролира само дали играта се провежда в момента. Той може да се използва за поставяне на пауза и когато играта завърши, за спиране на основната игра по време на екрана за приключване на играта.

И накрая, скриптът WaveSpawner също има проста цел - да следи на коя вълна се намира играчът и да създава враговете за тази вълна.

### Ui Скриптове

UiElement е скриптът, прикрепен към всеки елемент, който се вижда на екрана (всички кули и ъпгрейди). Най-важното е, че той съхранява какъв елемент е свързан с тази инстанция на UiElement. Той също така включва ToolTip, когато е поставен върху него.



*Фиг 6. показва UiElement инстанциран от ItemUiElement prefab-а за който е прикачен UiElement скрипта*

Ui скриптът на контролира частите на потребителския интерфейс {Фиг 7}, които са винаги на екрана, като количеството пари, с които разполага играчът, и инвентара на играча.

*Фиг 7. Потребителския интерфейс който Ui скрипта контролира*

Health скриптът контролира healthbarа, показан на {Фиг 8}

*Фиг 8.*

Скриптът ToolTip управлява поведението на Tooltip-a, когато той се постави върху различни неща, обикновено предмети в инвентара на играча или в прозореца за ъпгрейдване на кули.

*Фиг 9. Tooltip показва статистиките на кулата.*

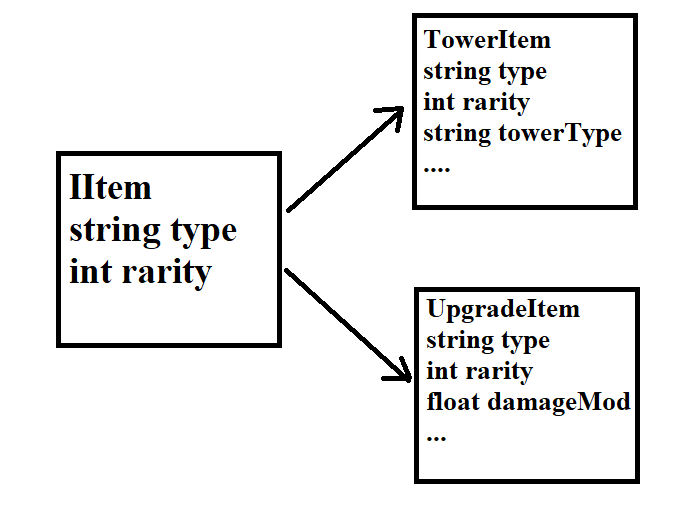
UiElementDragAndDrop и DropOnly обработват действието за влачене и пускане на елементи в инвентара и в други подобни екрани.



*Фиг. 10 Ъпгрейд в процеса на слагане на кула.*

### Item скриптове

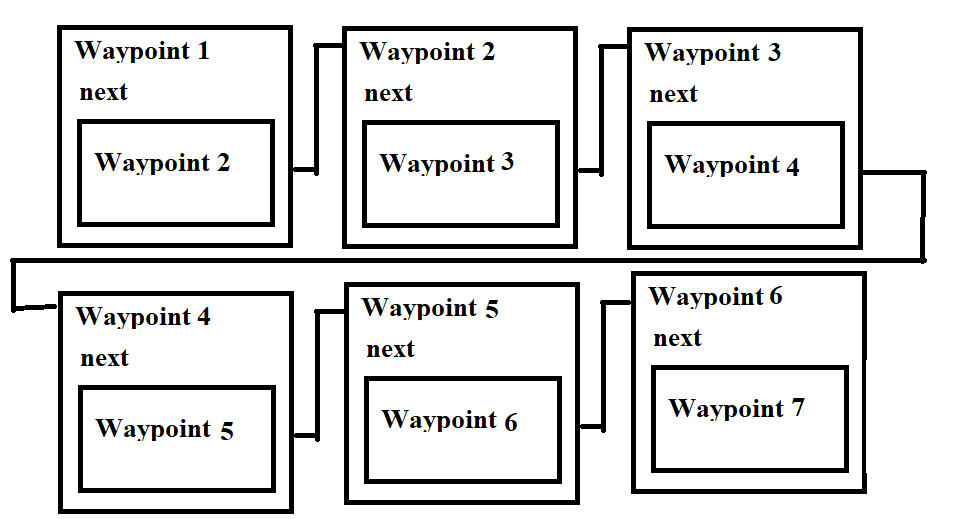
IItem е интерфейс, който се използва от класовете TowerItem и UpgradeItem. Това позволява и двата класа да бъдат поставени в една и съща колекция инвентар, така че не е необходимо да имам различни колекции за всеки от тях.



*Фиг. 11 показва връзката между IItem, TowerItem и UpgradeItem*

### Противници

Основният скрипт за врага заедно със скрипта за пътната точка работят заедно, за да създадат движение на врага, като врагът се движи към една пътна точка, а когато стигне до нея, получава препратка към следващата пътна точка по начин, подобен на linked list {Фиг. 12}. Когато скрипта не намери следващ waypoint приема че е стигнал края на пътеката и изчезва и сваля от живота на играча. От друга страна ако Живота на противника падне под или на нула пак изчезва, но този път увеличава на играча парите и има шанс да даде или угрейд или кула които се добавят към инвентара на базата. Всички променливи като например живот на противника, колко пари ще даде при смърт или колко от живота на играча ще вземе ако стигне края и дори какъв е шанса да остави кула/ъпгрейд след смъртта си биват запазеки в този скрипт.



*Фиг. 12 Връзката от един waypoint към следващия ползвана за движение на враговете*

# Тестване

Тествах играта си, като я изиграх. Всеки път, когато добавях функция, пусках играта, за да проверя как работи. По този начин индивидуално тествах всяка нова функционалност.

# Внедряване (deployment)

Кодът и изградената игра са качени в github. Ако този проект продължи да се разработва, той ще бъде пуснат или в itch.io, или в Steam, като и в двата случая ще бъде пусната само сглобената версия на играта без изходния код и файловете на Unity. При изтегляне начинът за достъп до основната потребителска част на проекта е чрез папката с играта и след това отваряне на .exe файла.

# Глава 3. Ръководство за потребителя

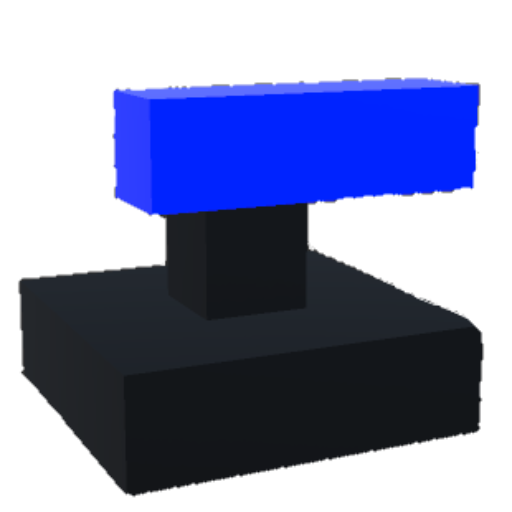
Проектът се съдържа в две папки "Игра" и "Код" . Папката "Game" съдържа експортираните файлове на играта, откъдето трябва само да стартирате .exe файла, за да играете играта. От друга страна, папката "Code", както подсказва името ѝ, съдържа Unity проекта с целия свързан код. За да отворите тази папка, ще ви трябват Unity и Visual Studio. Версията на Unity, използвана за този проект, е 2021.3.13f1, така че препоръчвам да инсталирате същата, за да предотвратите каквито и да е проблеми.

Ще запоnчна с кода, тъй като той ще бъде по-бързo. Всичко е разделено на папки. Повечето папки са генерирани автоматично и съдържат стандартни неща като библиотеки и пакети, които съм използвал. Нещата, по които съм работил, са в папката Assets. Файловете .meta се използват, за да кажат на Unity какво представлява всяка папка/файл/изображение и т.н. и как да ги използва.

" Icons " (Икони): Папката Icons съдържа спрайтове, използвани за потребителския интерфейс на играта.

"Materials": Папката ""Materials"" съдържа материали, използвани за обектите в играта.

"Prefabs" (Предварително подготвени елементи): папката Prefabs (Предварително подготвени елементи) съдържа prefab-ове. Prefabs е термин в Unity, който описва обект от играта, който може да се инстанцира с помощта на код. Използвам ги за неща като кули {Фиг. 13}, врагове и node-ове - все неща, които по принцип са почти еднакви всеки път.



*Фиг. 13 Кула Prefab*

" Scenes ": Папката "Сцени" съдържа сцените в играта. Те са само две: главното меню и сцената на нивото.

"Scripts": Папката "Скриптове" съдържа целия код, използван за функционирането на играта. След това всеки скрипт се присвоява на един или повече GameObjects в Unity, за да може играта да работи. Тази папка е разделена на следните подпапки, за да се улесни намирането на търсения скрипт

"ItemScripts" за скриптове, свързани с предмети,

"Managers" за скриптове, които имат само една инстанция в играта, обикновено не са прикрепени към действителен обект, те се използват за обработка на неща като строене, създаване на нивото и самия играч.

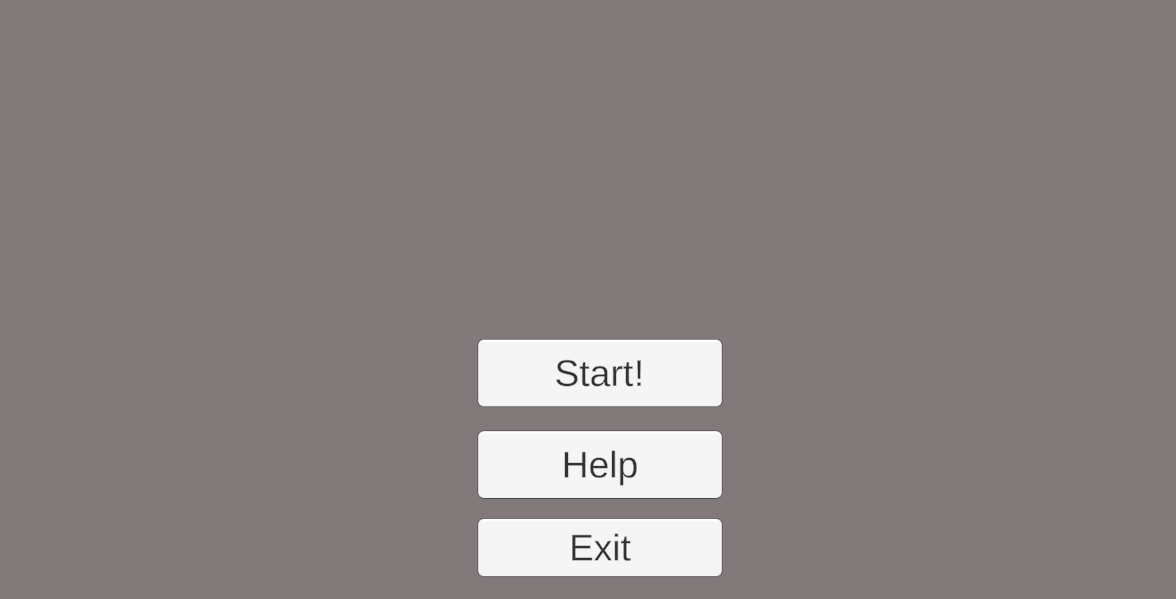
"Ui", разбира се, съдържа всички скриптове, свързани с потребителския интерфейс.

" Enemies " съдържа всички скриптове, свързани с враговете и поведението им.

Основната папка " Scripts" съдържа и скриптове, които не можах да групирам по никакъв смислен начин.

"TextMesh Pro": Тази папка е друга библиотека, използвана от Unity за обработка на всички текстове в играта.

Продължаваме с частта за играта. Когато стартирате играта, влизате в главното меню както е показано на {Фиг. 14}. Оттук имате възможност да кликнете върху "Старт", за да стартирате играта, върху "Помощ", за да научите как да играете играта, или върху "Изход", за да затворите играта.



*Фиг. 14 Главното меню*

### Играене на играта

Основният цикъл на играта се състои от следното:

1. поставяте кули

2. започвате вълна, която позволява на враговете да влязат в екрана

3. вашите кули убиват враговете

4. враговете пускат кули или ъпгрейдове

5. поставяте нови кули или подобрявате старите и повтаряте

### Обяснение на основния екран на играта{Фиг 15}:

В лявата част можете да видите текущото ниво.

Белите квадрати са възли с кули и можете да поставите по една кула на всеки от тях.

Тъмно сивите възли са пътят на враговете. Те ще вървят от червения куб в началото до зеления куб в края.

Зеленият куб е вашата база, като всеки път, когато враг изпусне предмет, той се поставя в инвентара на базата, който е различен от вашия инвентар.

От дясната страна можете да видите основната част на потребителския интерфейс. Горната част с всички кули е вашият инвентар. Това е мястото, където щраквате, за да изберете кула.

Под него вляво е живота ви. Той започва от 100 и намалява всеки път, когато враг премине през кулите ви. Когато стигне нула губите играта и трябва да започнете от началото.

Под живота е сумата пари, с която разполагате в момента.

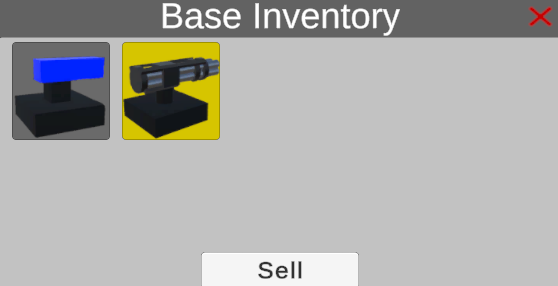
И накрая, вдясно от двете е бутонът за стартиране на следващата вълна. Той не може да бъде натиснат, докато в момента се създава вълна или ако враговете все още са на екрана.

### 

*Фиг. 15 Главния екран на играта*

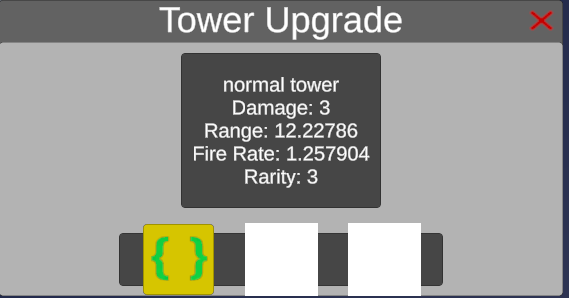
Можете да видите колко обекта има в базата от числото в горната ѝ част или ако поставите мишката върху нея, то ще се покаже като подсказка.

Ако щракнете върху базата, нейният инвентар ще се появи като прозорец {*Фиг. 16*}. Тук можете да проверите различните кули и подобрения и да ги добавите към собствения си инвентар, за да ги използвате, или да ги продадете за пари.



*Фиг 16. Инвентара на базата*

И накрая, ако кликнете върху вече поставена кула, можете да отворите прозореца за ъпгрейдване за нея{Фиг. 17}. Оттук можете да добавяте или премахвате ъпгрейдове. Когато добавите ъпгрейд към кулата, статистиките на кулата автоматично се актуализират. Освен това, ако поставите ъпгрейд върху друг ъпгрейд, вторият ще бъде поставен в слота, а първият ще бъде добавен в инвентара ви.



*Фиг. 17 Ъпгрейд екрана за кула*

# Заключение

Понастоящем проектът е завършена игра. Можете да правите всичко, което съм замислил да направя от самото начало като базова линия. Това не означава, че по нея не може да се направи повече. Ако продължа да работя по него, бих направил следните неща:

1. Наемане на дизайнер, който да създаде дизайн на различните кули, врагове и интерфейси. Сегашните дизайни са каквито и да е основни модели, които бих могъл да създам в Unity, но с помощта на Blender човек би могъл да направи много по-добре изглеждащи модели и вероятно дори да ги анимира.

2. Добавяне на по-голямо разнообразие от врагове и кули. С начина, по който е създаден проектът, това не би трябвало да представлява твърде голямо предизвикателство, тъй като те ще използват повечето от вече написания код за работа, като ще изискват само допълнителна работа, ако имат специфични функции, които не са създадени.

3. Публикувайте играта в Itch.io или Steam, за да може да достигне до повече хора. Това определено ще стане поне след като първата стъпка бъде завършена, така че играта наистина да изглежда добре.

4. Накрая бих накарал повече хора да опитат играта, за да видя дали другите я намират за интересна, тъй като имам пристрастно мнение по тази тема. След това да взема всички отзиви и да доразвия всички механики, докато повече хора не намират играта за приятна.

# Информационни източници

За източници използвах различни уроци от YouTube и различни публикации във форумите на Unity и StackOverflow.

# Рецензия на дипломен проект

| **Тема на дипломния проект** |  | | |
| --- | --- | --- | --- |
| **Ученик** |  | | |
| **Клас** |  | | |
| **Професия** |  | | |
| **Специалност** |  | | |
| **Ръководител- консултант** |  | | |
| **Рецензент** |  | | |
| **Критерии за допускане до защита на дипломен проект** | | **Да** | **Не** |
| Съответствие на съдържанието и точките от заданието | |  |  |
| Съответствие между тема и съдържание | |  |  |
| Спазване на препоръчителния обем на дипломния проект | |  |  |
| Спазване на изискванията за оформление на дипломния проект | |  |  |
| Готовност за защита на дипломния проект | |  |  |
| Силни страни на дипломния проект |  | | |
| Допуснати основни слабости |  | | |
| Въпроси и препоръки към дипломния проект |  | | |

**ЗАКЛЮЧЕНИЕ:**

Качествата на дипломния проект дават основание ученикът/ ученичката ................................................... да бъде допуснат/а до защита пред членовете на комисията за подготовка, провеждане и оценяване на изпит чрез защита на дипломен проект- част по теория на професията.

.........05.2023г. Рецензент:................................

град София