**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към**

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

**Тема: 2D платформър игра**

**Дипломант: Научен ръководител:**

*Боян Иванов инж. Марио Бенов*

**СОФИЯ**

2 0 2 1 г.

**УВОД**

В днешно време игрите са едно от най-разпространените средства за развлечение. Има различни жанрове игри, като всеки от тях е уникален и носи със себе си различни механики и стилове на игра. Те най-основно могат да бъдат разделени на 2D и 3D. 2D игрите имат множество подвидове, като един от най-популярните е платформъра. Името платформър идва от платформите, върху които играчът скача. Това позволява играчът освен напред да се движи нагоре и надолу, дава възможност за създаване на различни стилове игра, както и усещането за повече контрол върху героя. Освен платформите се използват различни препятствия и врагове, които играчът трябва да избегне, за да достигне до крайната цел.

Платформърите могат да се разделят на скролери, в които играчът обикновено стои в центъра на екрана, а фонът се движи зад него, или такива, при които фонът е статичен и играчът се движи из екрана. Освен това платформърите могат да бъдат шутър игри (shooter games), при които се използват оръжия, за да бъдат обезвредени враговете, или такива, в които играчът просто изследва света, като обикновено, за да атакува враговете, използва оръжия с малък обсег (мечове, ножове и др.) или скача върху тях.

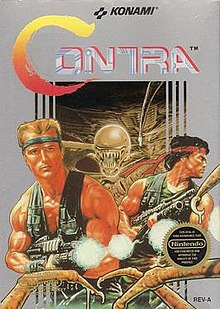
Целта на настоящата дипломната работа е да бъде създадена 2D платформър игра. Името на играта е Зона 52. В нея играчът може да използва оръжия, за да се справя с различни врагове. Също така е изправен пред предизвикателството да решава различни по сложност пъзели, за да прогресира в играта.

**ПЪРВА ГЛАВА**

**ПРЕГЛЕД НА ПОДОБНИ ДВУИЗМЕРНИ ИГРИ И РАЗВОЙНИ СРЕДИ ЗА РАЗРАБОТКАТА ИМ**

* 1. **Преглед на подобни двуизмерни игри**
     1. **Contra[2]**

Contra е run and gun видео игра, създадена от Konami, като първoначално е пусната като аркадна игра, а по-късно е интегрирана от Нинтендо за тяхната конзола.



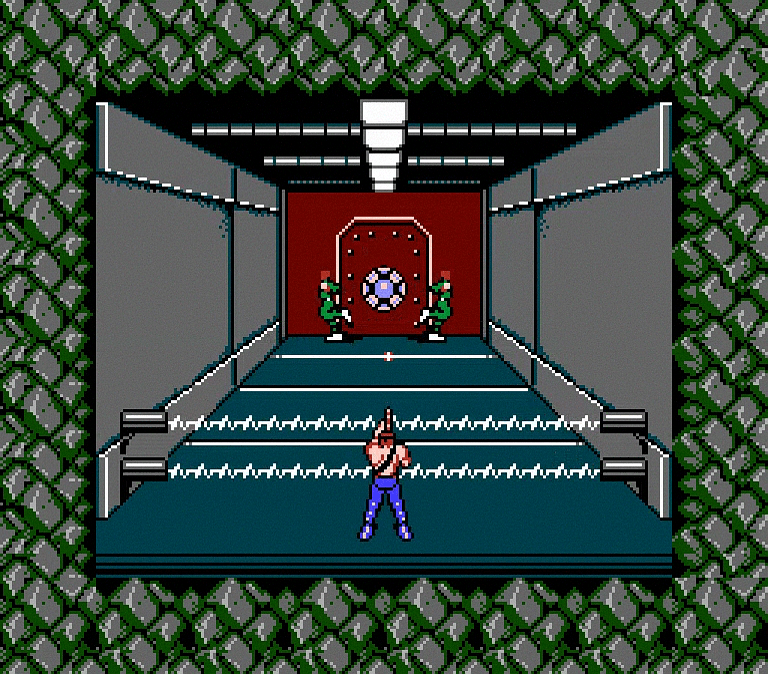
Фиг. 1.1 Корица на оригиналната Contra

Действието се развива през далечната 2633 г. Организацията „Червен сокол“ е създала база на архипелага Галуга близо до Нова Зеландия с цел унищожаване на човечеството. Двама командоси, Бил Райзър и Ланс Бийн от подразделението Contra на Earth Marine Corp (елитна група войници), са изпратени на острова, за да унищожат вражеските сили и да разкрият истинската същност на извънземното, което ги контролира.

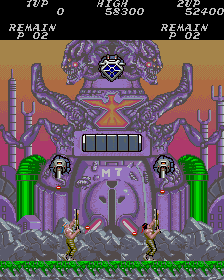
Contra използва различни перспективи, като стандартна странична перспектива (Фиг. 1.2), псевдо 3D, при която играчът се движи към фона, както и наляво и надясно, при което се създава илюзията за трето измерение, (Фиг. 1.3) и фиксиран екран, при който оръжието на играча е насочено постоянно нагоре (Фиг. 1.4).



Фиг. 1.2 Стандартна странична перспектива



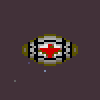
Фиг. 1.3 Псевдо 3D перспектива



Фиг. 1.4 Перспектива с фиксиран екран

Играта може да се играе от двама играчи, като всеки от тях избира героят от двамата командоси, който да бъде – Бил или Ланс. Контролите се състоят от 8-посочен джойстик и два бутона – един за скачане и един за стреляне. Когато играчът скочи, той прави салто, вместо стандартен скок, както при други игри. Джойстикът, освен движението наляво и надясно, контролира посоката на стреляне на оръжието. Когато джойстикът е натиснат надолу, играчът ляга на земята, което му позволява да избягва вражеските куршуми, както и да стреля по ниски мишени. Ако бутонът за скачане е натиснат, когато играчът е легнал на по-висока платформа, той ще слезе на долната такава.

Основното оръжие на играча е пушка с безкрайни амуниции, която може да бъде подобрена към други 4 оръжия. Това може да бъде постигнато чрез унищожаване на Pill Box Sensor (Фиг. 1.5) или летящите капсули (Фиг 1.6).

Фиг 1.5 Pill Box Sensor Фиг 1.6 Летящи капсули

Първата версия на Contra съдържа 7 етапа, които варират между различните перспективи. Играчът може да загуби живот, ако героят се докосне до враг или куршум, или падне в най-долната част на екрана, ако няма платформа, както и при свършване на времето. Когато това се случи, героят ще се върне към основното си оръжие и ще загуби един живот. При свършване животите на героя играта започва отначало. Вторият играч може да се присъедини по всяко едно време. Ако животите на единия играч свършат, той трябва да изчака другият също да изхаби своите животи, за да могат да продължат заедно.

Contra набира голяма популярност заради възможността за играене на двама играчи едновременно, което не е било разпространено при първоначалното излизане на играта. Тя става една от най-емблематичните игри в platform shooter жанра и става един от стандартите за създаването на такива игри занапред.

* + 1. **Broforce[3]**

Broforce е side-scrolling run-and-gun platform игра, създадена от Free Lives и публикувана от Devolver Digital през Октомври 2015г.



Фиг. 1.7 Лого на играта Broforce

Героите в играта са наречени bro (жаргонен израз за приятел). Основната им характеристика е, че са мускулести командоси, които спасяват заложници. Със спасяването на тези заложници се отключват нови Bros. Заобикалящата среда може да бъде разрушавана от оръжието на героя и от експлозиите на враговете.

Героите в играта са пародии на различни измислени екшън герой от известни филми, като Джон Рамбо, Джон Маклейн от Умирай Трудно (1988 г.), Чък Норис и други. Обикновено имената на героите са променени, като се слага думата „bro“ (например Rambro, the Brominator). Играчът няма контрол над това с кой герой ще започне нивото и при всяко спасяване на заложник или смърт на играча се сменя активният герой. Това създава динамика, защото всеки Bro има различни механики на игра, базирани на измисления герой, например героят на Елън Рипли използва огнехвъргачка, а този на Индиана Джоунс, камшик.

* + 1. **Разлики между посочените игри и дипломният проект**

Основната разлика между дипломния проект – Зона 52 и игрите Broforce и Contra е възможността за събиране на оръжия и предмети и наличието на инвентар. При Зона 52 играчът има опцията да взема предметите, които може да намери в различни сандъци или в труповете на противниците. След това тези предмети могат да бъдат продавани и могат да се закупуват други такива. Това липсва при другите две игри, тъй като те са насочени повече към стрелбата, отколкото към събирането и използването на различни предмети.

Друга съществена разлика е наличието на пъзел елементи под формата на мини-задачи. Пъзел елементите съществуват в играта Broforce, но те не са толкова силно изразени и се срещат само в няколко нива.

**1.2. Преглед на развойните средства за разработка на игри**

**1.2.1 Unity[4] и подобни среди за разработка на игри**

Unity е гейм енджин (game engine), предназначен за начинаещи разработчици на игри. Първоначално е обявен само за Mac OS X, но в момента поддържа повече от 25 платформи. Съдържа богат набор от инструменти за създаване и обработка на играта, анимиране, компилиране, дебъгване, оптимизация. Unity съдържа инструменти за обработка на 2D спрайтове, камери и светлини, симулация на физика (движение, маса, гравитация, сблъсъци и др.), обработка на аудио, създаване на анимации, възможност за съвместна работа между екипи и други. Основният недостатък на Unity е производителността, която е по-ниска от някои други гейм енджини.



Фиг 1.8 Лого на unity

Unity може да се използва за създаването на двуизмерни и триизмерни игри, както и за виртуална и добавена реалност. Основният език за програмиране е C#, като в предишни версии платформата е поддържала Boo и версия на JavaScript, наречена UnityScript.

Unity е безплатен за персонално използване, както и за малки фирми с приходи под $100 000 годишно. За по-големи фирми цената за използване е определена на базата на приходите, генерирани от играта.

Освен Unity съществуват още редица среди за разработка на игри. Една от най-известните е Unreal Engine. Използван е за създаването на множество игри като Fortnite, PlayerUnknown’s Battlegrounds, Batman: Arkham Knight и други. Той е безплатен за използване, като Epic Games, фирмата, която го е създала, може да вземе определен процент от печалбите на продукта. Unreal Engine e написан на езика C++, което му осигурява висока портативност.

Други подобни среди са Cryengine, Lumberyard, GameMaker, Godot, Ogre3D, RPG Maker, Marmalade и още десетки подобни.



Фиг 1.9 Лога на различни среди за разработка на игри

**1.2.2 Microsoft Visual Studio[5] и подобни развойни среди**

Популярна развойна среда, която има интеграция със Unity е Microsoft Visual Studio. Той е комерсиален продукт на Microsoft, който има безплатна Community версия, както и платени Professional и Enterprise версии, които включват по-напреднали инструменти за тестване и отстраняване на грешки, както и по-улеснена комуникация между програмистите. Въпреки че е безплатен, Community Edition може да се използва за комерсиални продукти.

Microsoft Visual Studio се използва за компилиране на C, C++, C# и Visual Basic код.



Фиг. 1.10 Лого на Microsoft Visual Studio

Unity има интеграция за Visual Studio, благодарение на която могат да се използват многобройни инструменти за писане, редактиране и дебъгване на код, както и помощни средства за форматиране, почистване, анализиране и рефакториране. Visual Studio поддържа функционалността IntelliSense (инструмент за автоматично завършване на код) и възможността за лесна навигация из кода.

Съществуват още редица други развойни средства за редактиране на код, като IntelliJ IDEA, Clion, Ecliplse, NetBeans и още много други, но те не поддържат толкова дълбока интеграция с Unity, колкото Visual Studio.

**1.2.3 Среди за създаване на 2D pixel art**

Pixel art е вид дигитално изкуство, при което изображенията се редактират на ниво пиксели. Този вид изображения произлиза от 8 и 16 битовите системи, които са били по-ограничени откъм производителност.

Съществуват много програми за създаване на Pixel Art, като всяка от тях осигурява различни функционалности.

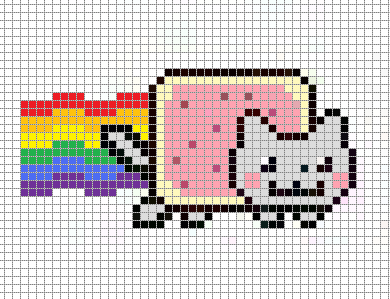
Най-известната сред тях е Adobe Photoshop[6]. Оригинално излязла през 1988, Photoshop става един от стандартите в индустрията за обработка на растерна графика. Инсталирането на т.н. plug-ins (допълнителни приставки), позволява гъвкавост и възможност за създаване и редактиране на множество типове изображения. Единственият недостатък е, че Photoshop е платена услуга. Като безплатна алтернатива може да се използва програмата Gimp, която поддържа по-малко функционалности.



Фиг 1.11 Лого на Adobe Photoshop

Друга не толкова популярна програма за създаване на pixel art е Pixel Art Studio. Тя е безплатна и лесна за използване програма, която е оптимизирана за работа с графичен таблет и устройства със touchscreen.

Освен нея съществуват още редица среди за създаване и обработка на 2D pixel art, като Aseprite, Pro Motion, Microsoft Paint, Krita, Pyxeledit и други.



Фиг 1.12 Пример за pixel art

**ВТОРА ГЛАВА**

**ФУНКЦИОНАЛНИ ИЗИСКВАНИЯ КЪМ ПРИЛОЖЕНИЕТО, АРГУМЕНТАЦИЯ НА ИЗБОРА НА РАЗВОЙНИТЕ СРЕДСТВА И СРЕДИ, ОПИСАНИЕ НА СЦЕНАРИЯ ЗА РЕАЛИЗАЦИЯТА НА ПРОДУКТА**

**2.1 Функционални изисквания към програмния продукт:**

* Проектът ще бъде изпълнен с Unity Game Engine, като среда за разработка и компилация. Кодът ще бъде написан на езика C# в средата Microsoft Visual Studio Community edition;
* Изображенията (sprites) ще бъдат нарисувани в Pixel Art Studio;
* Играта да се реализира в стил 2D platformer с един играч. Да съдържа две отделни нива, които демонстрират имплементираната функционалност;
* Реализиране на логика за управление на играча чрез клавиатура и мишка: движение, стреляне с екипирани оръжия, управление на инвентар;
* Реализиране на система "Инвентар" за съхранение и прехвърляне на предмети – оръжия, броня и пари. Съхранените предмети могат да се активират, използват, купуват или продават;
* Поддръжка на записване и продължаване на състоянието на играта;
* Реализиране на три различни вида противници, с различни свойства и методи на атака;
* Поддръжка на различни видове препятствия: мини пъзели, разрушаем терен, терен с негативни ефекти върху играча.

**2.2 Избор на език и развойна среда**

За среда е избрана Unity, защото е насочена към начинаещите в разработката на игри и в нея са базирани знанията, придобити в 11 клас от курса за Game Development. Използването на друга платформа би довело до затруднения и отделяне на допълнително време за разучаването ѝ.

За език за програмиране е избран C#, тъй като в Unity той е основният API за скриптовете. Освен това C# притежава известни прилики с изучавания по Обектно-ориентирано програмиране в 11 клас език Java.

За IDE е използван Microsoft Visual Studio заради удобствата, които създава интеграцията на Unity. Освен това Microsoft Visual Studio е най-разпространената развойна среда за разработка на C# програми и освен това предлага Community edition, който е безплатна версия на продукта.

За среда за създаване на 2D pixel art е избрана Pixel Art Studio, тъй като програмата е безплатна и предлага множество функции, като персонализиран размер на платното, поддръжка на множество слоеве, удобство при работа с графичен таблет и други.

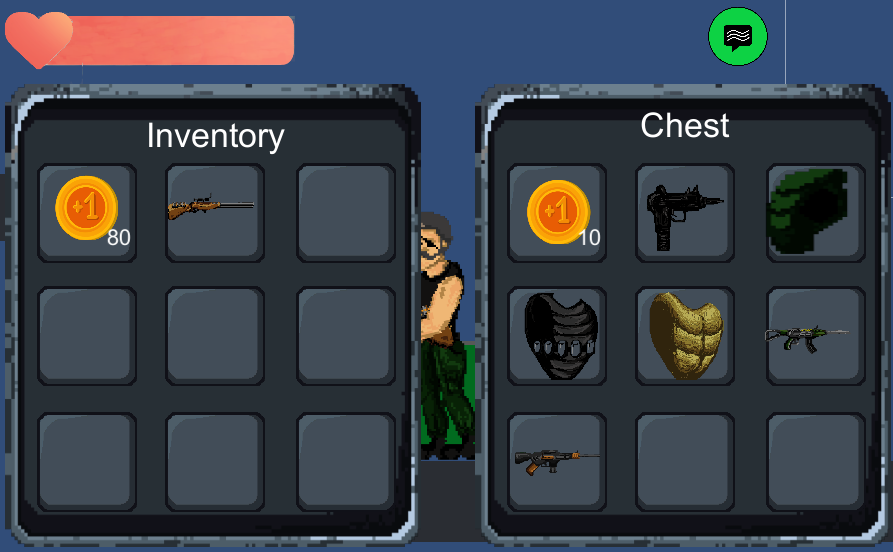
Освен това е предназначена за начинаещи в pixel art и не поддържа по-сложните функционалности на Photoshop, които намалят продуктивността на програмата.

**2.3 Описание на алгоритъма на работа на програмата.**

Приложението започва с инициализиране на играча. В скрипта Player.cs се зареждат данните за героя, като текущите екипирани предмети (оръжие, каска и броня), текущото ниво, до което е достигнал играча, инвентара на играча и други. След това заредените предмети се екипират. В скрипта PlayerInput.cs се изключват всички UI елементи, като инвентар, tooltip, магазини и т.н..

Освен това в началото на играта се инициализират всички обекти в сцената, като противници, препятствия, пъзели, помощни блокове, магазини, сандъци и платформи.

В UI\_Inventory.cs се зарежда инвентара, като се преминава през всеки елемент от него и се създава обект – itemSlotTemplate, който отговаря за визуализирането на този елемент в инвентара, като за всеки от тях се настройва количеството и иконата на предмета. Тези обекти се разпределят в инвентара и могат да бъдат премествани от играча.



Фиг. 2.1 Инвентар на играча, когато отвори сандък

Движението на играча се извършва в скриптa CharacterController2D.cs, като вземането на входните данни на играча се извършва в скрипта PlayerMovement.cs. Следенето на курсора на мишката и стрелянето на оръжието се извършва в PlayerAimWeapon.cs. Здравето на играча се следи и контролира в скрипта PlayerHealth.cs



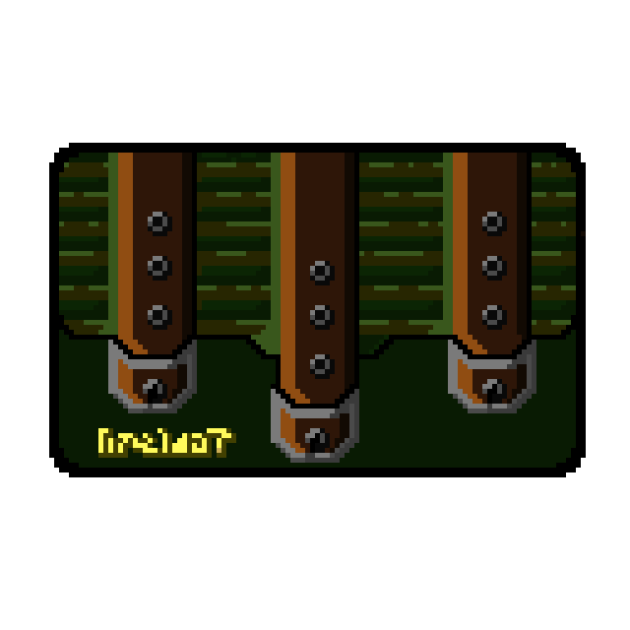
Фиг 2.2 Health bar

При задържане на курсора на мишката над някой предмет в инвентара се активира т.н. tooltip, който съдържа общи детайли за предмета.

****

Фиг 2.3 Примерен tooltip

Когато играчът се приближи до блок, с който може да си взаимодейства (сандък (Фиг 2.4), магазин, пъзел, труп на противник), в PlayerInput.cs при натискане на бутона „Е“ се активира UI елемент, който отговаря за съответният блок.



Фиг 2.4 Сандък, в който се съдържат предмети, които играчът може да вземе.

Друг тип обект е PickableCrate (Фиг 2.5), което е кутия, която може да бъде вдигана и премествана от играча. Тя може да служи като защита срещу куршуми, но може да бъде счупена при нанасяне на достатъчно щета. Тя се управлява от скриптът PickableBlock.cs.



Фиг 2.5 PickableCrate обект

Когато играчът не се намира близо до обект, за който има прикачен UI елемент, се активира менюто EquippedItems, което показва текущите броня, каска и оръжие. То се контролира от скрипта EquippedItems.cs



Фиг 2.6 Меню EquippedItems

Информацията за него се пази в PlayerData файла и се зарежда при началото на играта.

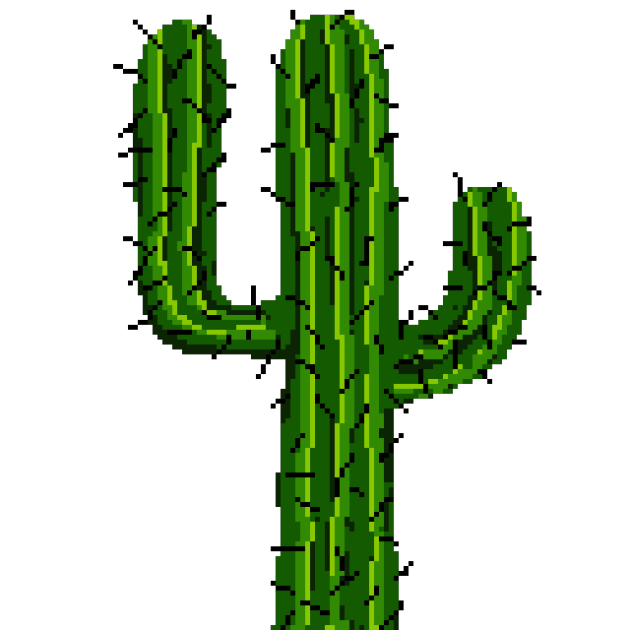
При натискане на бутона “P” на клавиатурата или на бутона за ръководство, който се намира на екрана, играта се паузира и се активира прозорец с подробна информация за оръжията и предметите.

При приключване на ниво се извиква функцията SavePlayer в Player.cs, която запазва прогресът на играча.

Играчът трябва да премине през различни препятствия. Те включват стена, която може да счупи (Фиг 2.7), която се управлява от скрипта BreakableWall.cs. Друг вид блок е DamageBlock, който при докосване от играча му нанася определена щета. Тя, както и скоростта на нанасяне на щетите, могат да бъдат регулирани чрез скрипта DamageBlock.cs. Обикновено този блок се среща в играта под формата на кактус (Фиг. 2.8), но освен това може да бъде срещнат и като блок за незабавно умиране, когато щетата е увеличена.



Фиг 2.7 Разрушима скала



Фиг 2.8 DamageBlock, от който при допир на играчът му се нанася щета.

Когато кръвта на играча свърши, той умира и се зарежда сцената GameOverScene. След като се натисне бутона Respawn се зарежда последното ниво, до което играчът е стигнал.

**ТРЕТА ГЛАВА**

**ПРОГРАМНА РЕАЛИЗАЦИЯ НА ДВУИЗМЕРНАТА ИГРА**

**3.1. Програмна реализация на играча**

Обектът на играча е съставен от седем скрипта:

* Player.cs
* PlayerInput.cs
* PlayerMovement.cs
* CharacterController2D.cs
* PlayerAimWeapon.cs
* PlayerHealth.cs
* InventoryObject.cs

Всеки един от тези скриптове отговаря за съответна функционалност на играча.

Player.cs е основният скрипт на играча. В него се съдържат основни функционалности като запазване и зареждане на играча, смяна на оръжия, брони и каски, както и вземането на кръв при нанасяне на щета на играча.

Запазването на прогреса на играча се извършва чрез класа SerializationManager. Той съдържа функция Save, която поема 2 аргумента: име на файла и обект, който да бъде запазен. В случая като обект подаваме класа SaveClass, който е Serializable. В него се съдържа обект от тип PlayerData. Класът PlayerData съдържа информация за играча като текуща кръв, която засега не се използва, текущите оръжие, броня и каска, името на текущото ниво и инвентара на играча.

Във функцията Save се проверява дали съществува папка „saves“ и ако не, се създава в Application.persistentDataPath директорията, която е директория за записване на информация между пусканията на играта. След това се създава файл със зададеното име и информацията се риализира и се записва във файла с помощта на BinaryFormatter. В класа на играча съществува функция SavePlayer, в която се създава и попълва обект от тип SaveClass и се извиква функцията Save от SerializationManager.cs.

По подобен начин работи зареждането на играча.

В SerializationManager.cs се съдържа функцията Load, която приема път към файла, от който искаме да зареждаме. В нея се проверява дали файла съществува и след това се десериaлизира и се връща обекта, който е получен. Основният клас Player съдържа функция LoadPlayer, който извиква горепосочената функция и задава данните от нея на текущия играч.

Скриптът PlayerInput.cs отговаря за UI елементите на екрана, като например инвентара, магазините и пъзелите. Идеята зад тези UI елементи е, че те са създадени предварително и се деактивират в началото на играта. На терена съществуват обекти, които репрезентират тези елементи, като сандъци, магазини, панели за пъзели и други. Тези обекти имат BoxCollider2D, на който е зададен да бъде Trigger. Така при засичането им от играча с функцията OnTriggerEnter2D, се променя стойността на boolean променлива, която отговаря за следния елемент. При натискане на бутона „Е“, се проверява кои променливи имат стойност true и се активират съответните им UI елементи. Ако вече има активирани елементи, което се пази в променливата isInventoryActive това означава, че играчът иска да затвори инвентара, тоест се деактивират всички UI елементи.

Скриптът PlayerMovement.cs отговаря за входните данни на играча и за изпълняването на анимациите му. Първоначално се проверява дали инвентарът на играча е включен, защото тогава движението трябва да бъде изключено. След това се проверяват бутоните за скачане, движение и клякане и се извикват съответните функции от класа CharacterController2D и се активират нужните анимации на играча.

Класът CharacterController2D[7] е взет от видеоклип на Brackeys[8] (канал в youtube) със заглавие “2D Movement in Unity (Tutorial)”[9] и е модифициран, така че да поддържа функционалности, необходими на програмата. Причината за вземането на кода е. че движението е най-важната част от проекта и добрата работа на кода е от голямо значение. Този скрипт позволява регулирането на височината на скока, скоростта на героя в клекнало състояние, изглаждане на движението и това дали героят може да бъде контролиран по време на скок. Скрипта е модифициран, така че да поддържа отделни colliders (колайдери) при състояния на скачане, клякане и ходене. Също така е добавено активиране на анимация за скок. Премахната е функцията Flip, тъй като завъртането на играча се контролира от мишката в друг скрипт. Функцията Move приема три аргумента. Число, което репрезентира скоростта на движение, като тя може да бъде между -1 и 1, и две Boolean променливи, които показват дали играчът скача или кляка. Тази функция контролира движението на играча, скоростта на движение в клекнало състояние и оставането му в това състояние след пускане на бутона за скачане при наличност на обект, който пречи на изправянето. Освен това тя отговаря и за активирането на различните колайдери, при различно действие (скачане, клякане, ходене), както и за самото скачане. Проверката за земя и таван се следи чрез два празни обекта m\_GroundCheck и m\_CeilingCheck. Проверката за земя се извършва и чрез LayerMask – m\_WhatIsGround, като се посочват кои слоеве от сцената се считат за земя.

Скриптът PlayerAimWeapon.cs отговаря за завъртането на ръката в посоката на курсора, завъртането на играча при достигане на определен ъгъл, както и стрелянето с оръжие. Първо във функцията HandleAiming се взема позицията на курсора и се изчислява посоката спрямо играча. След това чрез функцията Atan2 се взема ъгълът на завъртане на ръката и се преобразува в градуси. След това се променя ъгълът на Ойлер на ръката на играча, така че да сочи към курсора. Ако ъгълът е прекалено голям, играчът се завърта.

Функцията HandleShooting отговаря за стрелянето на играча през определен период от време. Използва се число – fireElapsedTime, което се увеличава в зависимост от това, колко време е минало. Ако това време стане равно на зададения от нас период на стреляне, стрелянето се позволява. Когато бутонът за стреляне е натиснат и стрелянето е позволено, се извиква функцията shoot на текущото оръжие и стрелянето отново се забранява.

Скриптът PlayerHealth.cs отговаря за количеството кръв на играча, както и за визуализирането му върху т.н. health bar. Класът съдържа няколко функции, които се грижат за това. Първата е reduceHealthPlayer, в която се изчислява щетата, която ще се нанесе на играча при удар, като се вземат предвид текущите броня и каска. След това променливата health, която следи кръвта на играча, се намалява и ако тя стане равна на 0, тогава играчът умира. Функцията reduceHealthEnemy се използва при враговете. За добавяне на кръв се използва функцията addHealth, като се посочва количество.

Класът InventoryObject съдържа текущият инвентар на играча. Той се състои единствено от променлива от тип Inventory. Това е така, тъй като класът Inventory трябва да бъде Serializable и не може да наследява MonoBehaviour, за да може да бъде сериализиран и запазван във файл. Класът Inventory се състои от списък от тип Item, който се казва inventory, както и функции за добавяне и премахване на Item. Функцията за добавяне AddItem проверява дали предметът е stackable, тоест дали количеството му може да е по-голямо от 1, и дали вече предметът съществува, след което се добавя към списъка.

Класът Item изглежда по следният начин:

//This class holds info for every item and is used in the Inventory class;

[Serializable]

public class Item

{

//A class for types of items and their amount;

public enum ItemType

{

Weapon,

Armor,

Helmet,

Coin,

}

private System.Random \_random = new System.Random();

private readonly float itemId;

public InventoryObject parrentInventory;

public string name;

public ItemType itemType;

public int amount;

public int buyPrice;

public int sellPrice;

public bool isStackable;

public Item()

{

itemId = \_random.Next();

}

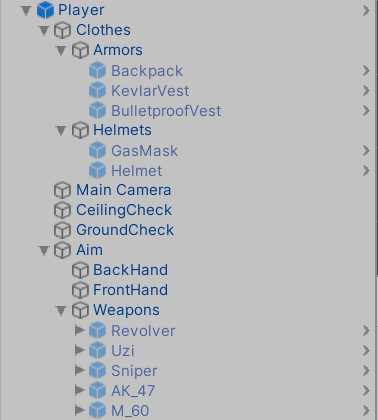
}

Той трябва да съдържа само данни, които са Serializable, т.е. класовете на Unity, като Sprite, Transform и други, не могат да бъдат използвани.

В него се съдържат следните променливи:

* itemId – Уникално число, по което се различават различните предмети. То се генерира чрез променливата \_random, която е от тип System.Random;
* parrentInventory – Съхранява инвентара, на който принадлежи предмета. Той е необходим, за да може при нужда предметът да бъде премахнат от него;
* name – Името на предмета освен за идентификация се използва и за задаване на Sprite от класа SpriteHolder;
* itemType – Съхранява типа предмет. Може да бъде оръжие, броня, каска или монета, като могат да бъдат добавени и още типове;
* amount – това е количеството на дадения предмет;
* buyPrice – цената, на която може да бъде закупен продуктът;
* sellPrice – цената, за която предметът може да бъде закупен;
* isStackable – определя дали предметът може да бъде съхраняван в количество, по-голямо от 1. Например оръжието може да бъде съхранявано само в единично количество, докато монетите могат да бъдат повече от една.

Оръжията, броните и каските на играча се съхраняват като отделни обекти, които са деактивирани при начало на играта и се активират, когато оръжието е екипирано.



Фиг. 3.1 Структура на играча

Обектът Aim играе ролята на ръце на играча. Той се завърта към курсора от скрипта PlayerAimWeapon.cs.

**3.2. Програмна реализация на враговете**

Съществуват три типа врагове – войници, статично оръдие и куче.

Войниците могат да бъдат статични и патрулиращи. Тяхното движение и стрелба се контролира от класа EnemyScript. В него се съдържат следните променливи:

// The point of the gun

public Transform firePoint;

// The bullet prefab that the enemy shoots

public GameObject BulletPrefab;

// The distance that the enemy can detect the player

public float distance;

// The gun of the enemy

public GameObject gun;

// The player so we can point to him

public Player player;

// The period between each shot

public float ShootingPeriod = 1f;

// The speed of movement

public float movingSpeed;

// The starting and ending point of the enemy

public Transform point1;

public Transform point2;

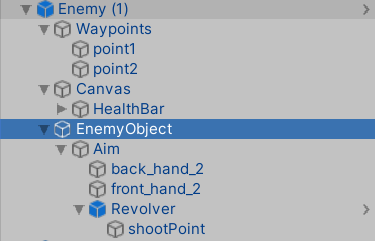
Всяка от тях отговаря за различна характеристика на врага и може да бъде променяна така, че да бъдат създавани различни по трудност врагове. Например ако не се зададат point1 и point2, войникът ще бъде статичен.

В Update метода се следи разстоянието на врага спрямо играча и в зависимост от това се извикват съответните функции, които отговарят за патрулирането и насочването на оръжието. Функцията Patrol отговаря за това врага да се движи между две зададени от нас точки в пространството. Проверява се посоката, към която сочи войникът, и след това се мести към желаната точка. След като я достигне, врагът се обръща и започва да се движи към другата точка.

Функцията HandleAiming отговаря за прицелването на врага към играча. Тя работи по подобен начин на HandleAiming в PlayerAimWeapon.cs, но вместо към курсора, целта е ръката на врага да се насочи към играча.

Стрелянето се контролира във функцията HandleShooting, като тя се извиква периодично през определен интервал от време – ShootingPeriod. В нея се проверява разстоянието до играча и ако то е по-малко от зададеното, се извиква функцията за стреляне на текущото оръжие. Когато врагът умре се извиква функцията die. В нея се задава колайдера на врага да бъде Trigger и се променя името на врага, като му се добавя думата „Chest“. Това се прави, за да може играчът да третира тялото на мъртвия противник като сандък и да може да се достъпи InventoryObject класа му. За визуализация се активира анимацията за смърт на противника.

EnemyScript.cs се задава на обект с име EnemyObject. Той съдържа в себе си обект Aim, който се състои от двете ръце и оръжието на противника. Освен скрипта на EnemyObject се задават още два скрипта – PlayerHealth, който отговаря за количеството кръв на врага и InventoryObject, който съдържа инвентара му. За health bar се използва отделен канвас и се слагат заедно с EnemyObject като деца на друг обект. Това се прави, за да може канваса да не наследява ротацията на обекта и когато врагът се завърти, да не се завърта и той.



Фиг. 3.2 Структура на враговете.

Вторият тип врагове са статичните оръдия. Те се управляват от скрипта TurretScript.cs и могат да се настройват със следните параметри:

public float Range;

public Transform Target;

public GameObject BulletPrefab;

public float angleOfView = 60;

public float damage = 10f;

public float bulletSpeed = 1f;

public float ShootingPeriod = 1f;

Променливата Range отговаря за обхвата на оръжието. Като Target трябва да се зададе играча. BulletPrefab е снарядът, който оръжието ще изстрелва. AngleOfView e максималният ъгъл, на който оръдието може да се завърти. Damage, bulletSpeed и shootingPeriod са характеристиките на оръжието при стрелба. Начинът на работа на скрипта е чрез следене на разстоянието от играча до оръдието. Когато това разстояние стане по-малко от зададеното, оръдието се завърта към играча и започва периодично да стреля по него.

Последният вид противник са кучетата. Те се контролират от скрипта Enemy\_behaviour.cs и съдържат следните характеристики:

// we need a refference to the player to flip the dog to his direction

public Transform player;

// this is the initial point for the raycast

public Transform rayCast;

// this is which layer the raycast is going to detect

public LayerMask raycastMask;

// the lenght of raycast( should be bigger than thе

collider size)

public float rayCastLenght;

// this is the distance at which the dog starts attacking

public float attackDistance;

// the speed of the dog

public float moveSpeed;

// a timer for the attack period of the dog

public float timer;

// how much damage per hit the dog deals

public float damage;

Кучето има три състояния – статично засичане, движение и нападане. При първото състояние кучето следи кога играчът ще попадне в triggerArea. Когато това се случи се изстрелва raycast, който е насочен към играча и кучето се приближава към него. Когато кучето стигне на определено разстояние от играча, то спира и започва да го атакува.

**3.3. Програмна реализация на UI елементите**

**3.3.1. Система „Инвентар“**

За да се визуализира инвентара, се използва скрипта UI\_Inventory.cs. Той се състои от панел, който служи за фон и деца, които визуализират всеки наличен предмет в инвентара. Използва се функцията RefreshInventoryItems, за да се обновяват елементите. В нея се обхожда инвентара и за всеки елемент се създава itemSlotTemplate, което е обект, който служи като шаблон. На всеки елемент му се задава клас DragDrop, който отговаря за това, той да може да бъде местен с курсора. Този клас имплементира интерфейсите IPointerDownHandler, IBeginDragHandler, IEndDragHandler, IDragHandler, IDropHandler, които му дават тази функционалност. В този клас също така се имплементират интерфейсите IPointerEnterHandler, IPointerExitHandler, чрез които е имплементиран tooltip. Чрез функцията OnDrop, която е от интерфейса IDropHandler, при поставяне на един елемент върху друг, се проверява, дали двата елемента са еднакви и дали са Stackable, и ако са такива, те се комбинират.

Под всеки инициализиран елемент се намира слот за поставяне на предмет. Ако над този слот има вече поставен предмет, той е зает и върху него не може да бъде сложен друг предмет, освен ако не е от същият тип и е Stackable. Тези слотове съдържат скрипта ItemSlot.cs, който имплементира интерфейса IdropHandler. Когато предмет бъде сложен отгоре, той се премахва от текущия си инвентар и като parrentInventory се задава текущият инвентар. След това предметът се центрира в слота и се проверява дали е еднакъв с предметите екипирани върху играча. Ако е еднакъв, това означава че той трябва да бъде премахнат от екипираните предмети.

**3.3.2. Магазини**

В играта съществуват магазини за купуване (Alien Store) и за продаване (Trash kings).

Магазинът за купуване се състои от слот за поставяне на монети, слот за получаване на предмет и набор от бутони, които се генерират и представляват елементите, които могат да бъдат закупени. Бутоните се активират при наличие на достатъчно монети поставени в слота и могат да бъдат натиснати. Функцията BuyItem се извиква, когато бъде натиснат бутон. В нея се създава нов Item, който се поставя в слота за получаване и се премахват толкова монети, колкото струва закупеният предмет.

Магазинът за продаване се състои от слот за предмет, който ще бъде продаден и слот за получените монети. Във функцията generateMoney при поставяне на предмет се проверява дали в слота за монети има нещо, и ако няма, се създават определен брой монети, които после могат да бъдат взети от играча. Ако вече има монети, просто се увеличава техният брой. След това се унищожава елементът, който е поставен.

**3.3.3. Панел за информация**

При натискане на бутона “P” от клавиатурата или бутона за ръководство, който се намира на екрана, се активира панела за информация, който съдържа информация за всички предмети в играта. Той използва скрипта ItemBookScript.cs, в който се съдържат два списъка – списък с имената (names) и списък с информация за всеки предмет (curriculum). Първоначално се създават първите две страници, които се визуализират, като се задава текста и изображението, свързано със съответния елемент. След това, в зависимост от натиснатият бутон, могат да се извикат функциите previousPage, която отваря предишната страница и nextPage, която отваря следващата.

В тях се сменят страниците, като се инкрементира брояча на текущата страница и се променя текста и снимката. Списъкът с информация се попълва с функцията fillCurriculum. При активиране на табелата играта се поставя на пауза.

**3.3.4. Пъзели и мини игри**

За отваряне на врата или спиране на определени препятствия се използват пъзели и мини игри. Това са панели с някаква задача, която трябва да бъде решена, като за момента са имплементирани три: ButtonPuzzle, CodePuzzle и HackingPuzzle.

ButtonPuzzle се състои от 5 превключвателя, всеки от които трябва да бъде включен, за да се реши задачата. Всеки превключвател съдържа скрипта SwitchScript, който следи неговото състояние и при натискането му го променя. Таблото се състои от изключени превключватели, върху които е поставено деактивирано изображение на включен такъв. По същият начин са направени и светлините, които отговарят за превключвателите. При включено състояние тези изображения се активират. Когато се активира, той се добавя към броя на включените превключватели и се изважда от него при деактивиране. Когато този брой стане равен на броя на превключвателите задачата е решена.

CodePuzzle се състои от поле за въвеждане на числа и бележка, на която е изписан кода, който трябва да бъде въведен. Този код се генерира автоматично. При промяна на стойността в полето за въвеждане, кодът се проверява и ако съвпада с генерирания, задачата е решена.

HackingPuzzle се състои от поле за въвеждане, конзола, табло с изписан текст “Access Denied” и бележка с командите, които трябва да бъдат въведени. В полето за въвеждане се проверява първата въведена команда. Ако тя съвпада, то се изчиства и командата се изписва в конзолата.

След това се проверява и втората команда. Ако и тя съвпада, задачата е решена и на таблото се изписва „Access Granted“, като цвета на текста се променя в зелено.

**3.3.5. Препятствия**

В играта са имплементирани няколко различни типа блокове, с които играчът може да си взаимодейства.

Класът BreakableWall се задава на всеки блок, който може да бъде разрушен. Той съдържа променлива, която следи кръвта на блока, както и лист от спрайтове, които са различните състояния на счупване на блока.

Класът PickableBlock се задава на всеки блок, който може да бъде повдигнат и носен от играча. При натискане на бутона “C” ако играчът се намира в trigger полето, позицията на блока започва да се променя спрямо тази на играча, като се добавя определено изместване.

Блокът за нанасяне на щета работи със скриптът DamageBlock.cs., като му се задава щета и интервал. След това при колизия на играча с този блок му се нанася щета и това действие се повтаря с горепосоченият интервал.

Освен този блок, съществува и такъв, който забавя играча. Той се контролира от скрипта SlimyBlock.cs и му се задава делител, с който се намаля скоростта на играча.

Последният вид имплементиран блок е движеща се платформа, която играе ролята на асансьор. Задават й се начална и крайна позиция, както и скорост на движение, след което се задава крайната позиция, като следваща и се задвижва към нея. След като стигне, като следваща, се задава началната позиция и така се редуват.

Всички тези блокове могат да бъдат спрени, защото наследяват скрипта StoppableObject, който съдържа променливата isEnabled и функцията changeState, която я променя.

**ЧЕТВЪРТА ГЛАВА**

**РЪКОВОДСТВО НА ПОТРЕБИТЕЛЯ**

**4.1. Инсталация и изисквания към компютърната конфигурация**

За да бъде пусната играта, трябва да бъде отворено приложението „Area52.exe“. Таргетирана е към платформата Windows, като препоръчителните изисквания са:

* **CPU**: Intel Core i3-4160
* **CPU SPEED**: 2.0 GHz
* **RAM**: 2 GB
* **OS**: Windows 7 64-bit SP1
* **VIDEO CARD**: DX10 compatible or better
* **PIXEL SHADER**: 4.0
* **VERTEX SHADER**: 4.0
* **FREE DISK SPACE**: 300 MB

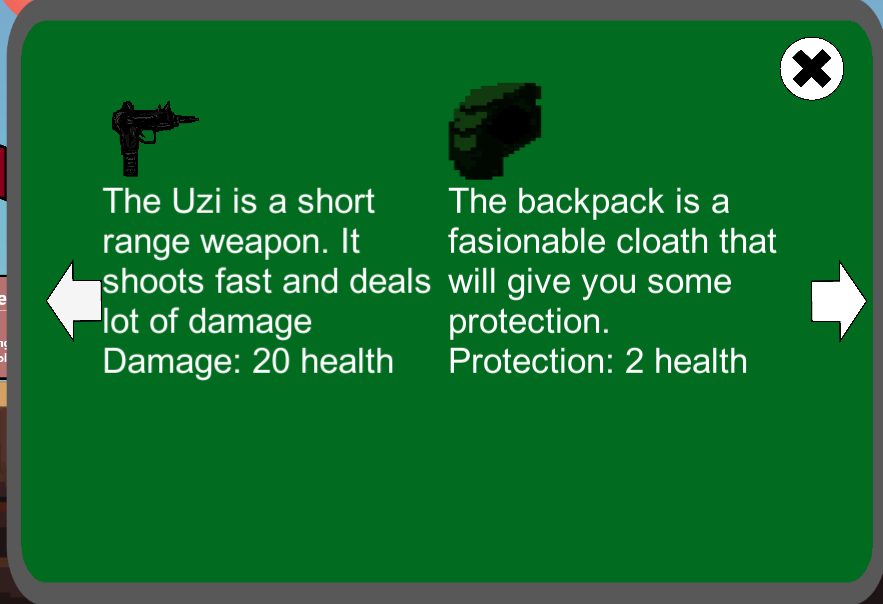
**4.2. Описание на продукта**

При стартиране на играта, се зарежда първото ниво. За контролиране на играча се използва мишка и клавиатура. Оръжието винаги сочи към курсора и може да бъде използвано с натискане на левият бутон на мишката.



Фиг 4.1 Сцена при стартиране на играта

В горния ляв ъгъл на екрана (Фиг 4.1) се намира т.н. health bar(Фиг 2.2). Той отчита текущото количество кръв, с което разполага играча. Вдясно от него се намира бутона за ръководство, който при натискане активира панел с информация за предметите в играта (Фиг 4.2). Алтернатива за включване на този панел е чрез бутона “P” от клавиатурата.



Фиг. 4.2 Панел за информация

При активирането на този панел играта се паузира. С двете стрелки може да се преминава през различните страници, а с натискане на бутона „X“, панела се затваря.

При натискане на бутона „Е“ се отваря инвентара на играча (Фиг 4.3). Отляво се намират предметите, които играчът притежава. Те могат да бъдат премествани, като бъдат завлечени с мишката, а при задържане на курсора, върху тях се появява т.н. tooltip (Фиг 2.3), който показва повече информация за предмета.



Фиг. 4.3 Инвентар

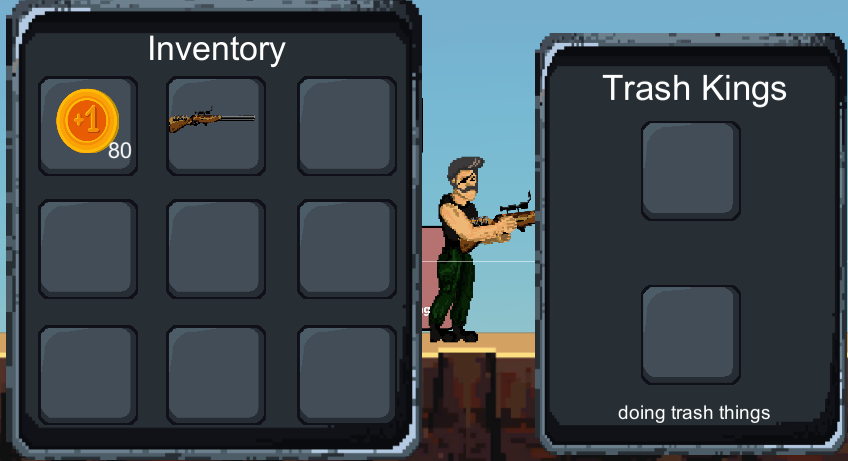
Отдясно се намират 3 слота. Първият е предназначен за различни видове каски, вторият за брони, а третият за оръжия. В тях не могат да бъдат слагани други видове предмети. След като бъдат сложени там, те се екипират върху играча.

Играчът може да съхранява своите предмети в сандъци (Фиг 2.4). След като се приближи до тях и натисне бутонът „Е“ се отваря менюто на сандъка (Фиг 4.4). Отляво се намира инвентара на играча, а отдясно се намира съдържанието на сандъка.



Фиг. 4.4 Прозорец, изобразяващ предметите в сандък

Предметите могат да бъдат продавани в замяна на монети в магазина „Trash Kings“. След като играчът се приближи до него, може да натисне бутонът „E“, при което се появява прозореца на магазина. (Фиг 4.5)



Фиг. 4.5 Прозорец на магазина “Trash Kings”

Вляво е инвентара на играча, а вдясно има два слота. В горния може да бъде поставен предмет за продажба. При поставяне, предмета изчезва и в долния слот се появяват толкова монети, колкото е стойността на продадения предмет.

По подобен начин се отваря и магазина за купуване на предмети „Alien Store“(Фиг 4.6). При него отново вляво се намира инвентара на играча, а вдясно, се намират поле за поставяне на монети, предмети, които могат да бъдат закупени и поле за вземане на закупения предмет. При поставяне на определен брой монети в горното поле, се активира бутона за закупуване на предмет. Когато този бутон се натисне, поставените монети изчезват и закупеният предмет се появява в съответното поле. След това този предмет може да бъде взет и поставен в инвентара.



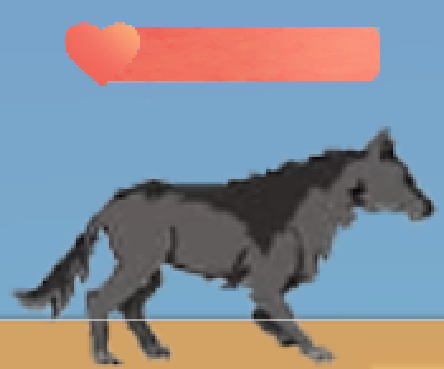
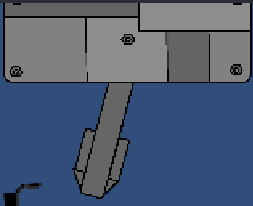
Фиг. 4.5 Прозорец на магазина “Alien Store”

Играта съдържа 3 вида противници. Първият е войник (Фиг 4.6). Той може да е статичен или патрулиращ. Когато играчът влезе в неговото полезрение, той започва да го стреля. Различните противници могат да имат различни оръжия и характеристики, съответно различни нива на трудност. Над противника се намира неговият health bar. Когато кръвта му свърши, той умира и играчът може да отиде до трупът му и с бутонът „Е“ да вземе предметите, които има.



Фиг 4.6 Войник

Вторият вид противници са кучетата. (Фиг 4.7). Когато играчът се приближи до него, то започва да го преследва. След като го настигне, започва да го атакува през определен интервал от време. Третият вид противник е статичното оръдие (turret) (Фиг. 4.8). След навлизане в полето на засичане, оръдието се завърта към играча и започва да стреля по него през определен интервал от време. Той не може да бъде убит от играча, но може да бъде спрян чрез решаване на даден пъзел.

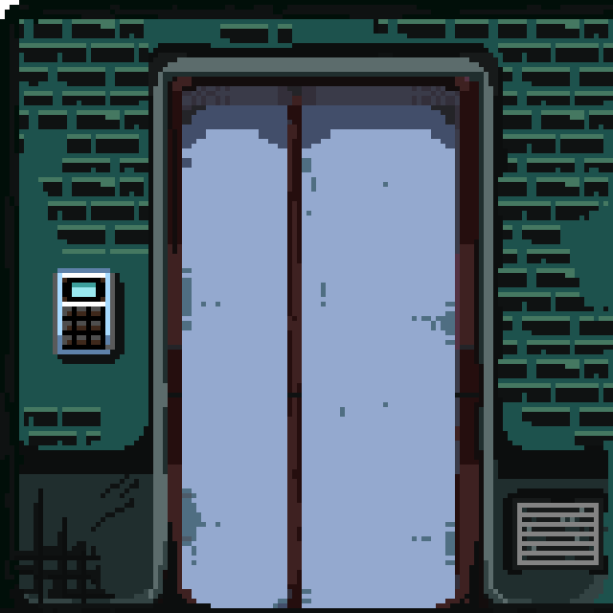
 

Фиг. 4.7 Куче Фиг. 4.8 Оръдие

Съществуват три вида пъзели под формата на мини игри. Те могат да се достъпят при приближаване до съответния панел с натискане на бутона „Е“. Първият вид е пъзел, при който всички превключватели трябва да бъдат включени. При втория, трябва да се въведе комбинация от числа, записани на бележка и залепени върху панела. Последният е с най-високо ниво на сложност .При него трябва да се въведат следните две команди:

* gcc hack\_program.c
* ./a.out

С тяхна помощ могат да се отключват врати (Фиг. 4.9). Освен заключени, вратите могат да бъдат и отключени, при които, за да се отворят трябва да се натисне бутона „F“.

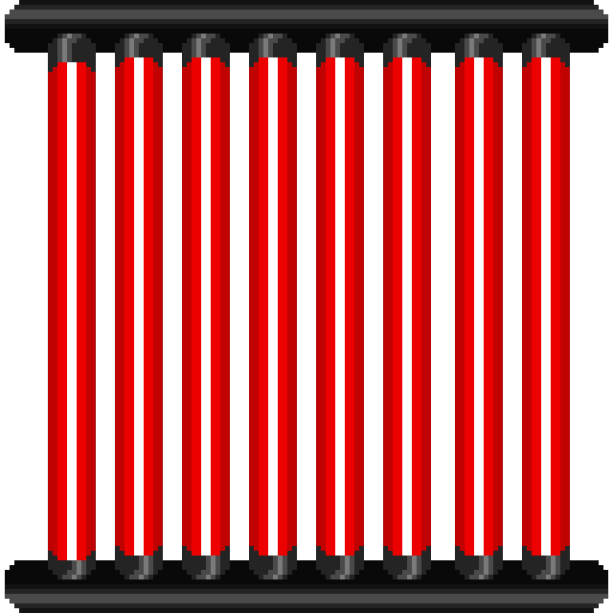


Фиг. 4.9 Врата

Играта съдържа много и различни препятствия. Едно от тях е т.н. Pickable Crate (Фиг. 2.5). Това е кутия, която може да бъде повдигана, при натискане на бутона „C“. Тя може да бъде използвана за защита от противниците, но при нанасяне на достатъчно щета, тя може да се счупи.

Друг вид препятствие е скалата (Фиг. 2.7). Тя е обект, който трябва да бъде счупен за да може да бъде преминато през него.

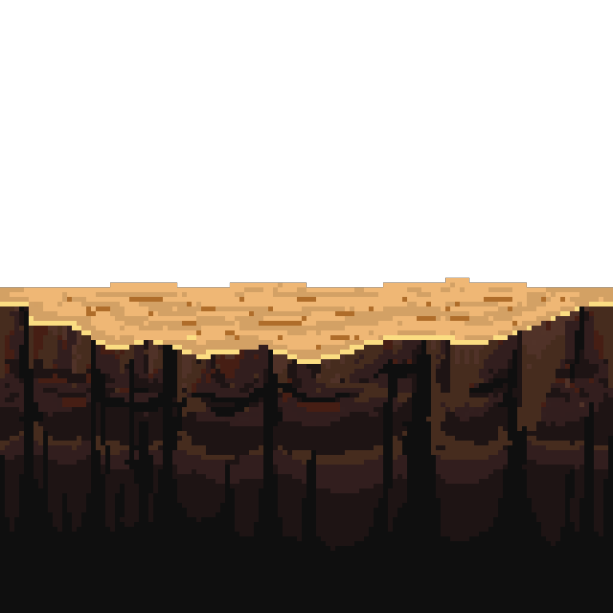
В играта има и три блока с негативни ефекти върху играча. Първият е кактуса (Фиг 2.8), който нанася щета при докосване през определен период от време. Вторият са лазерите. (Фиг 4.10) Те, подобно на оръдието могат да бъдат изключени чрез решаването на пъзел.



Фиг. 4.10 Лазери

Третият и последен е дълбокият пясък. (Фиг. 4.11) При него скоростта на играча се намалява.

Освен това има и блок асансьор, който се движи между две точки. Той може да пъде пускан и спиран чрез бутон или пъзел.



Фиг. 4.11 Дълбок пясък

При намаляване на кръвта, играчът може да се излекува в т.н. Health Station. Това се случва при приближаване и натискане на бутона „H“. Ако кръвта на играча свърши, той умира и е съживен в началото на нивото, като се губи прогреса, който е направил през него.

**ЗАКЛЮЧЕНИЕ**

В дипломната работа са постигнати следните цели:

* Реализиране на базова логика на играча: движение с помощта на клавиатура и мишка; стреляне с екипирани оръжия; умиране на играча и загуба на играта, когато му се нанесат достатъчно щети;
* Реализиране на система "Инвентар" за съхранение на предмети – оръжия, броня и пари. Съхранените предмети могат да се екипират върху играча или да се използват за търгуване;
* Поддръжа се записване и продължаване на състоянието на играта;
* Реализиране на система за купуване и продаване на предмети в замяна на пари;
* Реализирани са пет различни вида оръжия, два вида каски и три вида брони;
* Реализирани са три противника с различни свойства и методи на атака и с възможност за промяна на характеристиките им. Те носят в себе си различни предмети, които могат да бъдат взети от играча след смъртта им;
* Поддържат се три мини игри;
* Поддръжка на разрушаем терен;
* Реализирани са три различни блока, с ефекти върху играча;
* Реализиран е панел за информация.

Идеи за бъдещо развитие:

* Добавяне на повече карти/нива, които следват определена история;
* Добавяне на повече оръжия, брони и каски, както и добавяне на предмети, които могат да бъдат използвани от играча;
* Добавяне на различни видове противници;
* Добавяне на меню с настройки за играта.

**ИЗПОЛЗВАНА ЛИТЕРАТУРА**

1. Видове 2D игри - <https://amazinggameroom.com/video-games/types-of-2d-games/>
2. Contra - <https://en.wikipedia.org/wiki/Contra_(video_game)>
3. Broforce - <https://en.wikipedia.org/wiki/Broforce>
4. Unity - <https://en.wikipedia.org/wiki/Unity_(game_engine)>
5. Microsoft Visual Studio - <http://www.microsoft.com/visualstudio/en-us/>
6. Adobe Photoshop - <https://www.adobe.com/bg/products/photoshop.html>
7. CharacterController2D.cs - [https://bit.ly/2MQAkmu](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqblh5NzJ4ZC0yWDZPT3JkN2szSU9iSWlRR0c4UXxBQ3Jtc0tuMVU2NVlhRHNtdmd6bXJrMEczS0oyUEg3eFJQVmVLUENCMjlRQ2RhVG13WElnQ1RUWFpwdmVkMEZvemZ2cmdpMnpvb3BZLUtRLUlkZkQ5Vjh1LTd3OWU2clVrWm9RcldwU0xISFgxYVBqSUdXQVc0SQ&q=https%3A%2F%2Fbit.ly%2F2MQAkmu)
8. Brackeys <https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA>
9. 2D Movement in Unity (Tutorial) - <https://youtu.be/dwcT-Dch0bA>
10. How to Equip Weapons and Items in Unity (Drag Drop, Inventory) - <https://youtu.be/isAmoM3RPEI>
11. Simple Inventory System in Unity (Store, Use, Stack and Drop Items) - <https://youtu.be/2WnAOV7nHW0>
12. Aim at Mouse in Unity 2D (Shoot Weapon, Unity Tutorial for Beginners) <https://youtu.be/fuGQFdhSPg4>
13. How to Build A Save System in Unity - <https://youtu.be/5roZtuqZyuw>
14. How to make a Tooltip (Unity Tutorial for Beginners) - <https://youtu.be/d_qk7egZ8_c>
15. 2D ENEMY Melee Combat in UNITY - Coding Attack Pattern - Part 2 - <https://youtu.be/waj6i9cQ6rM>

**СЪДЪРЖАНИЕ**

**Увод**..................................................................................................................2

1. **Първа глава:** Преглед на подобни двуизмерни игри и развойни среди за разработката им...........................................................................3
   1. Преглед на подобни двуизмерни игри...............................................3
      1. Contra...........................................................................................3
      2. Broforce........................................................................................6
      3. Разлики между посочените игри и дипломният проект ....................................................................................7
   2. Преглед на развойните средства за разработка на игри..................................................................................................8
      1. Unity и подобни среди за разработка на игри.........................................................................................8
      2. Мicrosoft Visual Studio и подобни развойни среди................10
      3. Среди за създаване на 2D pixel art................................................................................................11
2. **Втора глава:** Функционални изисквания към приложението, аргументация на избора на развойните средства и среди, описание на сценария за реализацията на продукта..................................................13
   1. Функционални изисквания към програмния продукт................................................................................................13
   2. Избор на език и развойна среда........................................................14
   3. Описание на алгоритъма на работа на програмата.........................14
3. **Трета глава:** Програмна реализация на двуизмерната игра...............20
   1. Програмна реализация на играча.....................................................20
   2. Програмна реализация на враговете................................................26
   3. Програмна реализация на UI елементите........................................31
      1. Система „Инвентар“.................................................................31
      2. Магазини...................................................................................32
      3. Панел за информация...............................................................32
      4. Пъзели и мини игри..................................................................33
      5. Препятствия..............................................................................34
4. **Четвърта глава:** Ръководство на потребителя....................................35
   1. Инсталация и изисквания към компютърната конфигурация.......35
   2. Описание на продукта.......................................................................35

**Заключение**..................................................................................................43

**Използвана литература**.......................................................................................44

**Съдържание**...........................................................................................................45