**Міністерство освіти і науки україни**

**державний університет телекомунікацій**

**кафедра Інженерії програмного забезпечення**

**ЗВІт**

по виконанню проекту:

з предмету розробка комп’ютерних ігор

«The-Dispare»

студента 3 курсу групи ПД-31

кафедри інженерії програмного забезпечення

Сабадах В.С

Викладач

аспірант, асистент кафедри

інженерії програмного забезпечення,

Дібрівний О.А

оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

-Київ 2018-

**Зміст**

**Unity**

**1)Що таке Юнити**

Unity — багатоплатформовий інструмент для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосування працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360.

Є можливість створювати інтернет-додатки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосування, створені за допомогою Unity, підтримують DirectX та OpenGL.

Рендеринг:

Графічний рушій використовує DirectX (Windows), OpenGL (Mac, Windows, Linux), OpenGL ES (Android, iOS), та спеціальне власне API для Wii. Також підтримуються bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), динамічні тіні з використанням shadow maps, render-to-texture та повноекранні ефекти post-processing.

Unity підтримує файли 3ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks та Allegorithmic Substance. В ігровий проект Unity можна імпортувати об'єкти цих програм та робити налаштовування за допомогою графічного інтерфейсу.

Для написання шейдерів використовується ShaderLab, що підтримує шейдерні програми написані на GLSL або Cg. Шейдер може включати декілька варіантів реалізації, що дозволяє Unity визначати найкращий варіант для конкретної відеокарти.Unity також має вбудовану підтримку фізичного рушія Nvidia PhysX (колишнього Ageia), підтримку симуляції одягу в системі реального часу на довільній та прив'язаній полігональній сітці (починаючи з Unity 3.0), підтримку системи ray casts та шарів зіткнення.

**2)Чому саме Юнити**

Анімації, були включені для скелетних мешів, також можна імпортувати. У Unity вони прикріплюються до імпортованого об'єкту, як кліпи анімації (animation clips), а в UE4 називаються послідовностями анімації (animation sequences). У першому руху управляються за допомогою контролерів анімації (animation controllers), а в другому за тим же принципом діють анімаційні Blueprints.

В обох двигунах є стейт-машини, що визначають переходи з одного стану Ассет в інше. У UE4 система називається Persona, а в Unity - Mecanim. Також можливе застосування скелетних мешів одного скелета до інших, але в Unity це в основному використовується для анімації гуманоїдів.

У UE4 анімації можна редагувати, в Unity - практично немає, особливо погано справа йде з рухами гуманоїдів. На думку автора, движки не підходять для професійного анімації персонажів - краще використовувати програми на кшталт Blender або Maya, а результат імпортувати в вигляді. Прикріплений до об'єктів матеріал додається в проект, але його властивості на зразок шейдера або текстур доведеться застосовувати вручну.

Для цього в Unity потрібно задати матеріалу шейдер і додати в його слоти текстури - карти шорсткостей, нормалей або дифузії. Власні шейдери доведеться писати самостійно або за допомогою сторонніх інструментів на кшталт Shader Forge або ASE. А в UE4 вбудований дуже потужний редактор матеріалів, заснований, як і система Blueprints, на нодах.

Для програмування в UE4 використовується мова C ++, який не всі люблять через складність і тривалість компіляції. Однак Jayanam вважає, що у движка зрозумілий API і прийнятний період компіляції. У UE4 дуже потужна і опрацьована система візуального скріптованія - Blueprints, за допомогою якої можна досягти практично тих же результатів, що і c C ++.

Unity 5 підтримує мови C # і UnityScript. API і його концепт дуже схожий на аналог з UE4. При використанні керованого мови на зразок C #, програміст не зобов'язаний використовувати покажчики (pointers), компіляція відбувається швидко.

У UE4 вбудований постпроцесінг. До сцени можна застосовувати bloom-ефект, тонування і антиалиасинг як глобально, так і до окремих її частин (за допомогою компонента PostProcessVolume).

У Unity є стек постпроцесингу, який можна завантажити з магазину Ассет движка. Система менш гнучка, ніж в UE4 - ефекти застосовуються тільки стеком або скриптами до камери.

**Поняття відеоігор**

**1)Що таке відеогра**

Відеогра — це електронна гра, в ігровому процесі якої гравець використовує інтерфейс користувача, щоб отримати зворотну інформацію з відеопристрою. Електронні пристрої, які використовуються для того щоб грати, називаються ігровими платформами. Наприклад, до таких платформ належать персональний комп'ютер та гральна консоль. Пристрій введення, який використовується для керування грою, називається ігровим контролером. Це може бути, наприклад, джойстик, клавіатура та мишка, геймпад або сенсорний екран.

Терміном «платформа» називають комбінацію апаратного та програмного забезпечення, на якому здатна працювати гра. Також використовується термін «система».

Платформи зазвичай поділяються на дві основні категорії: персональні комп'ютери (ПК), які слугують для різних цілей, серед яких і можливість грати у відеоігри, та гральні консолі — пристрої, створені саме для ігор. Консолі в свою чергу поділяються на стаціонарні, які підключаються до телевізора чи монітора, і портативні, які гравець може носити з собою.

Окремо виділяють аркадні ігрові автомати та мобільні пристрої, як смартфони чи планшетні комп'ютери, на яких також можна грати у відеоігри.

**2)Жанр(схожі ігри, недоліки та достатки)**

Також існує багато жанрів відеоігор але ми вирішили обрати для свого проекту жанр «Survival horror» тому що він є цікавий в самій розробці та викликає приплив адреналіну у гравців.

Недоліки такого жанру – це прісність сюжету, передбачуваність ходу гри та маленький вибір варіацій геймплею.Достатки такого жанру – це напружена атмосфера протягом всієї гри, виклик своєму страху та зрозумілість механіки гри.

**Планування**

**1)Планування гри(Ми запланували таку гру такого жанру ляляля)**

Перед початком планування проекту, ми надихалися такими схожими по жанру іграми як: Resident Evil, The Evil Within 2

Ці ігри мають добре розроблені елементи несподіваності та напруженості.

Тому ми вибрали жанр «Survival horror» в якому ми вирішили спертися на напружену та захоплюючу атмосферу гри.

**Сюжет**

Сюжет гри розгортається в місті (назва) наповненим живими мерцями, в якому виявляється головний герой після аварії автобуса який перевозив його до в'язниці після неправдивого обвинувачення

**2)Задачи(План роботи)**

**1)Створити повністю іграбельного персонажа(Ходьба, керування камерою, різновид зброї, анімації)**

**2)Створити зомбі(Штучний інтелект, різноманітні звуки під різні дії, анімації)**

**3)Створити коробки з патронами для зброї**

**4)Збереження та завантаження прогресу**

**5)Левел дизайн**

**6)Меню паузи у ігровій сцені**

**7)Головне меню**

**Ролі у розробці**

**1)Владислав Сабадах: Головний аніматор, кодер.**

**2)Євгеній Сєчко: Помічник головного аніматора та кодера.**

**3)Марія Лотош: Головний композитор та 3Д артист.**

**Розробка ігрової сцени**

За нашою ідеєю нам були потрібні патрони для зброї, які гравець мав здобувати самотужки, тому ми вигадали ідею з коробками, в яких знаходились патрони, також створили анімацію перезарядки зброї.

**BoxHP.cs(Скрипт, який дає змогу знищити коробки з патронами)**

|  |  |
| --- | --- |
| public float Health; | **Поле з кількістю життя коробки** |
| public SphereCollider ammoTrigger; | **Тріггер коллайдер, який з’являєтся при знищенні коробки щоб мати змогу підібрати патрони** |

Персонажу якось треба отримувати патрони для зброї? Так, але як це зробити? Легко, можно створити коробки з боєприпасами, з яких можно буде дістати патрони для зброї.

**AmmoBox.cs**

|  |  |
| --- | --- |
| public int bulletInBox; | **Публічне поле, яке дорівнює кількості патронів, які випадуть з коробки** |
| public BulletsScipt bltScript; | **Поле яке дає дозвіл керувати скриптом з іншого скрипта** |
| public Text ammoText; | **Текст, який висвічуєтся коли персонаж має змогу підібрати патрони** |
| public Text ammoPickedText; | **Текст, який показує кількість піднятих патронів** |
| public Animator ammoTextAnim; | **Аніматор в якому є анімація тексту при підбиранні патронів** |
| public bool pickupFlag=false; | **Булевий флаг, який дозволяє або не дозволяє підбирати патрони в залежності від того чи розбитий ящик чи ні** |

Коробки з патронами є, але як дати персонажу якусь саме кількість патронів? Для цього ми створили скрипт, який можна змінювати для будь-яких коробок на сцені.

**BulletsScript.cs(Скрипт, з логікою патронів при перезарядці та пострілах)**

|  |  |
| --- | --- |
| public int BulletsInClip = 8; | **Поле з кількістю патронів у обоймі** |
| public int BulletsLeft = 16; | **Поле з кількістю патронів всього** |
| public Text bulletsUI; | **Текст, який показує на екрані гравця патрони в магазині та скільки всього** |
| public Animator reloadAnim; | **Аніматор з анімацією перезарядки** |
| public ShotAllow shootScript; | **Доступ до скрипта, який дає дозвіл на постріл** |

Як же без правильної логіки патронів? Що коли вони закінчатся, а пістолет все одно буде стріляти? Саме тому ми зробили скрипт з логікою для стрільби та перезарядки пістолета

**BackScript.cs(Скрипт, який закриває меню налаштувань і відкриває меню паузи)**

|  |  |
| --- | --- |
| public GameObject menu; | **Canvas з меню** |
| public GameObject settings; | **Canvas з налаштуваннями** |

Для гри дуже важливим являється пункт з меню, адже це важливо мати доступ до певних налаштувань, тому ми зробили меню старту гри, з налаштуваннями та меню паузи.

**ContinueButtonGame.cs(Скрипт для виходу з паузи ”При натисканні Continue вимикаємо меню, вмикаемо скрипти пересування/повороти камерою/дозволу стрільби”)**

|  |  |
| --- | --- |
| public MouseLook mouseLook; | **Скрипт керування камерою** |
| public ShotAllow shoot; | **Скрипт дозволу пострілів** |
| public FirstPersonControllerFix movement; | **Скрипт пересування** |
| public GameObject menu; | **Canvas меню** |

Доступ до меню та меню паузи це звичайно чудово, але потрібен і вихід з цього меню, потому ці скрипти создані для повертання гравця в ігровий процес.

**SetSensValue.cs(Збереження та зміна чутливості миші)**

|  |  |
| --- | --- |
| public Slider musicSlider; | **Слайдер, в якому змінюємо значення чутливості** |
| public MouseLook mouseSens; | **Доступ до скрипта, для зміни значення чутливості** |
| public Text musicText; | **Текст зі значенням чутливості** |

Від чутливості миші безпосередньо залежить ефективність в грі, тому це дуже важливий критерій для гравця, саме тому ми створили налаштування чутливості.

**GunShotScript.cs(Логіка пострілів та їх анімації)**

|  |  |
| --- | --- |
| public GameObject bulletDecal; | **Об’єкт декали, який спавнится при попаданні по об’ектам** |
| public ParticleSystem flash; | **Партиклі вогню при пострілі** |
| public ShotAllow shootFlag; | **Скрипт дозволу пострілу** |
| public Animator cameraShootAnim; | **Анімація віддачі від зброї** |

Який шутер без зброї та пострілів? Ми створили скрипт, який дозволяє гравцю вправлятися з зброєю, перезаряджати її та стріляти з неї.

**HandsKick.cs(Логіка ударів руками та їх анімації)**

|  |  |
| --- | --- |
| public Animator handsKickAnim; | **Анімація ударів** |
| public GameObject AttackCollider; | **Активація коллайдера при ударах** |

Не завжди у персонажа будуть патрони для знищування зомбі, тому ми зробили удари руками щоб розбивати ящики з патронами для пістолету.

**Load.cs(Завантаження останнього збереження)**

|  |  |
| --- | --- |
| public Transform Cur\_play\_position; | **Поле з позицією персонажа для завантаження** |
| public BulletsScipt bulletsScipt; | **Поле з кількістю патронів для завантаження** |
| public save\_script save; | **Скрипт збереження для отримання данних персонажа** |
| Animator Zombie\_dead; | **Анімація смерті зомбі після завантаження сцени** |
|  |  |

В таких іграх дуже складно без збереження, потому ми створили змогу гравців зберігати свій прогрес та завантажувати.

**PauseScript.cs(Скрипт для паузи на ігровій сцені)**

|  |  |
| --- | --- |
| public bool pause = false; | **Поле «Флаг» для перевірки паузи** |
| public FirstPersonControllerFix movement; | **Вимикання скрипта руху при паузі** |
| public ShotAllow shoot; | **Вимикання скрипта пострілів при паузі** |
| public GameObject menu; | **Вмикання Canvas меню паузи** |

Нам треба було зробити так щоб було можливо в будь-який момент призупинити гру, тому ми створили цей скрипт, який в самій грі відповідає за вимикання ігрового процесу та переходу до меню паузи.

**RecoilInRun.cs(Заборона віддачи пострілу при бізі)**

|  |  |
| --- | --- |
| public Animator reloadAnim; | **Аніматор з анімацією перезарядки** |
| public FirstPersonControllerFix moveScript; | **Скрипт, з яким ми забороняемо віддачу пострілу при бізі** |

Не можливо стріляти у ціль коли персонаж біжить, адже адже грає анімація бігу, саме тому ми заборонили віддачу постріла при бізі.

**ReloadSounds.cs(Звуки перезарядки зброї)**

|  |  |
| --- | --- |
| public AudioClip[] ShotClips; | **Кліпи зі звуками пострілів** |
| public AudioSource audio; | **Аудиоджерело для програвання пострілів** |

Дуже атмосферно, коли ти граєш і чуєш свої постріли, саме тому ми додали звуковий супровід.

**RunToIdle.cs(Преривання бігу при пострілі)**

|  |  |
| --- | --- |
| public Animator animator; | **Аніматор з анімацією бігу** |
| public bool runInAnim=false; | **Флаг дозволу бігу** |
| public FirstPersonControllerFix allowRuning; | **Перемикання бігу в ходьбу** |

Нам потрібно було перервати біг для пострілу, для цього ми перемикаємо з бігу на ходьбу при пострілі.

**SetMusicVolume.cs(Збереження та зміна гучності звуків)**

|  |  |
| --- | --- |
| public Slider musicSlider; | **Слайдер, в якому змінюємо гучність** |
| public AudioSource musicSource; | **Аудіоджерело відтворювання звуків, в якому змінюємо гучність** |
| public Text musicText; | **Текст зі значенням гучності** |

Нам потрібну було зробити гру так, щоб кожен гравець міг її налаштувати під себе, бо не для кожного підходить якась певна гучність.

**SettingsActive.cs(Вимикаємо меню паузи\головне меню при натисканні кнопки Settings)**

|  |  |
| --- | --- |
| public GameObject menu; | **Вимикає Canvas меню** |
| public GameObject settings; | **Вмикає Canvas Settings** |

Цей скрипт був створений для переходу між головним меню та меню налаштувань.

**ShootFlash.cs(Вмикає івент партиклів при пострілі)**

|  |  |
| --- | --- |
| public ParticleSystem flash; | **То вмикаем партиклі** |
| public ShotAllow shootFlag; | **Якщо постріли дозволені** |

Для більшої реалістичності ми вмикаємо івент партиклів, що задає атмосферності грі, адже «Survival horror» має за головну перевагу саме атмосферність.

**ShotAllow.cs(2 Методи, які використовуються як івенти в анімаціях, які дозволяють або забороняють стріляти)**

|  |  |
| --- | --- |
| public bool shoot; | void ShootAllowness()    {      shoot=true;    }    void ShootDeniedness()    {      shoot=false;    } |

Нам треба було дозволяти в певних ситуаціях вмикати, або вимикати дозвіл на зброю, наприклад коли персонаж перезаряджаєтеся або біжить.

**WeaponSwitch.cs(Скрипт для зміни зброї)**

|  |  |
| --- | --- |
| public GameObject LeftHand; | **При натисканні клавіши «1» дістаєтеся ліва рука** |
| public GameObject RightHand; | **При натисканні клавіши «1» дістаєтеся права рука** |
| public GameObject Pistol; | **При натисканні клавіши «2» дістаєтеся пістолет** |
| public AudioClip[] ShotClips; | **Кліпи звуків при доставанні зброї** |
| public AudioSource audio; | **Аудіоджерело для програвання цих звуків** |

В нашій грі ми додали 2 вида зброї, кожний гравець може вибрати те що більше підходить до ситуації, та змінити зброю завдяки цьому.

**WoodKickParticleScript.cs(Скрипт для програвання партиклів дерева при ударі в коробку)**

|  |  |
| --- | --- |
| public GameObject woodParticle; | **Об’єкт партиклів** |
| public GameObject myCamera; | **До якого об'єкта повинні ідти партиклі** |

Для більшої реалістичності при ударі ми додали партиклі дерева, це надає реалістичності грі.

**WoodShotParticleScript.cs(Скрипт для програвання партиклів дерева при ударі в коробку та нанесення шкоди зомбі та програвання партиклів крові)**

|  |  |
| --- | --- |
| public GameObject bloodParticle; | **Партиклі крові** |
| public GameObject woodParticle; | **Партиклі дерева** |
| public GameObject myCamera; | **До якого об'єкта повинні ідти партиклі** |
| public BulletsScipt bletScript; | **Скрипт для перевірки патронів(якщо більше 0)** |
| public ShotAllow shootScript; | **Скрипт для перевірки можно стріляти** |
| public float weaponDamages = 30f; | **Поле з уроном, який наносить зброя** |

Для нас дуже важлива атмосферність, тому ми так же додали партиклі крові при нанесення шкоди мерцям.

**ZombieHP.cs(Здоров’я зомбі та сейв його позиції, якщо здоров’я менше 0)**

|  |  |
| --- | --- |
| public float zombieHealth; | **Поле зі здоров'єм зомбі** |
| public GameObject Zombie; | **Об'єкт зомбі** |
| public save\_script save; | **Скрипт для зберігання статусу та позиції** |
| Animator zombieAnimator; | **Анимація зомбі при смерті** |

Ми створили опонентів, тому їх треба було налаштувати для боротьби с з нами, задати їм здоров’я а також змогу зберігати їх статус.

**ZombieTarget.cs(Зомбі іде до цілі, якщо персонаж в зоні його пристуності)**

|  |  |
| --- | --- |
| public NavMeshAgent zombieNavMesh; | **Штучний інтелект зомбі для переміщення** |
| public Transform player; | **Позиция головного персонажа** |
| public Animator zombieAnim; | **Анімації зомбі при дії середовища на нього** |

Штучний інтелект дуже важлива частина в цій грі, адже без опонентів в неї нудно грати, а зомбі задають атмосферу, а також є перешкодами на шляху гравця. Це гра на виживання, тому треба вижити.

**Розробка сцени меню**

**Тестування проекту**

**Висновок**

Метою роботи було розробити повноцінну гру, яка допомагає користувачеві гри зануриться в похмуру атмосферу.

В ході виконання роботи були реалізовані всі поставлені завдання:

1. Створення ландшафту в ігровому світі. Були детально розібрані основні моменти при створенні ландшафтів в Unity.
2. Створення контролера для персонажа. Контролер допомагає нам пересуватися в нашому ігровому світі, після натискання клавіш WASD. А за допомогою лівої кнопки миші персонаж має можливість зробити атаку з обраного зброї.
3. Реалізація пошуку найменшого шляху і коректне пересування NPC по ньому.

Вся розробка велася в вельми популярному ігровому

движку Unity3D. Так як в подальшому це буде Survival Horror, то для неї

потрібно вдосконалити графіку і музичний супровід.

Даний проект є тільки малою частиною повноцінної гри. адже

для того, щоб створити повноцінну гру, необхідний ряд додаткових

робіт, які пов'язані з сюжетом, геймплеєм і іншими речами.

**1)Список лит**