Софийски университет „Св. Климент Охридски”

Факултет по математика и информатика

Катедра „Софтуерни технологии“

Дипломна работа

Изследване на методи за адаптиране на видео игри за стрелци от първо лице спрямо кривата на учене на играча

Дипломант: Илко Захариев Адамов
Специалност: Софтуерни Технологии
Факултетен номер: 25394

Научен ръководител:
проф. д-р Боян Бончев

гр. София, 2018 г.

Съдържание

1. Увод

1.1. Актуалност на проблема и мотивацията

В света на компютърните игри съществуват доста жанрове който за различните хора представляват различен интерес (стратегически игри, екшън игри, ролева игра, приключенска игра, симулативни игри, спортни игри, сериозни игри). За голяма част от хората компютърните игри са част от тяхното свободно време, забавлявайки се с любимата си игра. В днешно време технологиите претърпяват сериозно развитие което се отразява и върху развитието на компютърните игри. Голяма част от игрите са ориентирани към самия играч, поддържайки интереса към любимата му игра по няколко начина, различните нива на трудности, възможност за игра в multiplayer режим, историята на играта, звуковите и визуалните ефекти в играта и др. Игровия жанр върху който ще се спрем е „Сериозни игри“. Сериозна игра е общо название за игра, която е проектирана с цел различна от забавление. Произлиза от сферата на компютърните образователни игри, но с течение на времето се развива и покрива много широк спектър от приложения с игрови елемент. Много често сериозните игри се разглеждат заедно със симулациите, като пример за сериозни игри могат да бъдат военни, диагностични и др. [1].

Сериозните игри са широко разпространени в определени области на обучението, където практическото обучение в реални условия е свързано с рискове – медицинско обучение, управление на комплексни машини и др. Сериозните игри заимстват от всички игрови жанрове. Най-популярни са рекламните игри и детските образователни игри, за професионално обучение широко са се наложили пилотните симулации, военни и медицински обучения. Въпреки че най-често сериозните игри се разглеждат в контекста на симулативните игри, ние ще се спрем на игра от тип стрелба от първо лице (екшън от първо лице). При този тип игри гледната точка на играчът съвпада с тази на компютърния герой, като от последния се вижда най-много ръката или ръцете държаща/щи оръжието. Този тип интерфейс е най-вече удобен за огнестрелни оръжия, защото насочването прицелването е сравнително лесно с мишката като устройство за управление. Тези игри разполагат с определен арсенал от мощни оръжия, които героя намира в различни нива или се играе multiplayer (което е истинското предизвикателство за играча). При игровия процес се отчитат попаденията тест елиминирането като точки известни като Праг, пример за такива игри са Call of Duty 2, Doom 3, Halo и др. [2]. В последно време подобен тип игри претърпяват сериозни промени, правейки ги много атрактивни за играча, поставяйки го един свят подобен на реалния или един напълно измислен свят (Call of Duty – главния герой е поставен в един свят подобен на реалния по времето на втората световна война, Doom – героят е поставен в един напълно измислен свят). Въпреки, че този тип игри са претърпели сериозно развитие, остава си проблема с адаптивността на играта спрямо новото на играча. Най-често проблема се решава в самото начало на играта където самия играч задава нивото на трудност избирайки степента преди да започне да играе. По този начин играта решава с колко противника ще се сблъска играчът, какво ще е поведението на заложения в логиката на играта изкуствения интелект (AI – artificial intelligence), как и в кои ситуации ще се промени средата около героя, количеството точки необходими за достигане на крайната цел на играта и др. Проблемът с адаптивността на играта може да се реши като спрямо модела на играча включващ в себе си моментните резултати в определен времеви диапазон, динамично се променя трудността на играта или казано по друг начин, адаптиране характеристиките на играта спрямо играча. Един от основните методи за адаптация се

основава на зададени нива на метрики на играча като при достигането им или надвишаването им се променя и игровия процес.
Освен по този начин, адаптивността на играта може да се базира на конкретен тип крива на учене описана със съответните метрики, като самата крива може да е линейна, експоненциална и др. като разпознаването се осъществява с помощта на шаблони. За целта на изследването на описаните методи, ще бъде разработена игра със стрелба по случайно движещи се обекти от първо лице (Box Shooter) в три версии, като първата версия е неадаптивна версия, втората е адаптивна версия с класическия метод за адаптиране при достигане на определени прагови нива и третата версия ще използва метода с откриване на кривата на учене [3] [4] [5] [6] [7]. За по достоверни резултати играта в трите и версии ще бъде предоставена на случайни хора. Освен това всеки от тях ще бъде анкетирани, като по този начин ще се определи до колко предложените методи са били полезни. Подобен тип изследвания могат да се използват и за бъдещи разработки на игри (в частност сериозни игри) който да стоят в основата на образователните игри, военни, диагностични и др. давайки възможност на играча да придобие нужния опит, умения и знания.

1.2. Цел на задачите на дипломната работа

Целта на дипломната работа е да се изследват методите за адаптиране на 3D видеоигра за стрелци от първо лице. Тя е разработена в рамките на научно-изследователския проект APOGEE (http://apogee.online), който планира да създаде решение, което от една страна да запълни липсата на проста, но ефективна софтуерна платформа с отворен код за лесно изграждане на образователни видеоигри от не- ИТ специалист, а от друга страна – да отговори на нуждите от огромно количество специализирани адаптивни за обучение по различни учебни дисциплини. Първият метод който ще използваме е класическия метод за адаптиране, който е базиран върху задаване на граници на резултата постигнат от играча и промяна на игровия процес (увеличаване скоростта с която се движат обектите, добавяне на мъгла, добавяне на дъжд, промяна на видимостта и размера на обектите, промяна на количеството типове обекти по които играчът стреля) при достигането на тези граници. Друг метод, който ще бъде изследван е откриване на кривата на учене, описана от метрики за резултата и разпозната в реално време чрез анализ на шаблони, спрямо която ще се извърши динамичната адаптация на сложността на играта. В допълнение към това ще се извършат игрови сесии и ще се добави анкета с помощта на която тези методи за адаптация на видео игри от подобен тип, ще се валидират. Основните задачи произлизащи от изследването на тези методи за адаптация на подобен тип игри са следните:

- Анализ на характеристиките на видеоигрите от този тип (мъгла, дъжд, ясна видимост, размер на обектите, количеството типове обекти, скоростта с която се движат обектите) спрямо модела на играча.
- Анализ на технологиите и софтуерните платформи необходими за разработка на игри от тип екшън от първо лице. Тази задача е особено значение тъй като играта трябва да върви бързо, да може да се играе освен в offline режим (на стационарен компютър или лаптоп без интернет връзка), така и в online режим (на стационарен компютър или лаптоп с интернет връзка).
- Анализ и проектиране в три различни версии на online 3D видеоигра от тип стрелба от първо лице, като първата версия няма да има адаптивни компоненти

променящи характеристиките на игровия процес. Втората версия на играта ще е с компонент за откриване на криви на учене на играча и адаптиране на характеристиките на игровия процес според тези криви. Третата версия на играта ще е с компонент за следене на резултата на играча спрямо който ще се адаптират характеристиките на игровия процес спрямо достигнатия резултат.

- Количествено изследване за определяне крива на учене на играча, на базата на която да се заложи адаптивност на динамична сложност на играта по избрани характеристики за трудност.
- Чрез анкетно проучване и игрови сесии се извършва количествено изледване на практическо валидиране на ефективността на вече създадените методи за адаптацията на видеоиграта.
- Анализ на получените резултати и различията м/у подходите за адаптация, чрез криви на учене, по нива, без адаптация, и да се направят изводи за приложимостта им във игри от този тип.

1.3. Очаквани ползи от реализацията

В областта на видеоигрите както при тези за развлечение, така и при сериозните игри почти не се използва динамично адаптиране на трудността на играта. С прилагането на различните методи на адаптиране играчът може да изпита по добро преживяване докато играе. Също така би подобрило неговите умения и опит в променящата се среда спрямо това как се справя с поставените му задачи и препятствия.

Компонентът за адаптация спрямо откритите криви на учене в една игра, ще може да бъде лесно използвана и внедрен във всяка една игра. Което от своя страна не само ще я подобри но и би станала по привлекателна и интересна за самите играчи, което би довело до повече хора които да си закупят играта следователно би донесло по-големи печалби за компанията разработила играта.

При сериозните игри които са ориентирани да помагат на ученици и студенти, една такава адаптация би била добър стимул да придобият нужните знания докато се забавляват. При симулатори, които имат за цел да тренират хората използващи ги за свършване на дадена работа по добре и успешно, такава адаптация би ги подготвила за всяка една възникнала ситуация. При хората с увреждане подобна игра симулация с динамично адаптиране би подобрило тяхното състояние и когнитивни умения.

1.4. Структура на дипломната работа

Дипломната работа е разделена на отделни теми, които са оформени като глави.

Първата глава въвежда читателя в проблема и мотивацията за изследването на методите за адаптиране на игра от първо лице спрямо кривата на учене. Описани са основната цел и произтичащите от нея задачи, очакваните ползи от реализацията

Втора глава е посветена с запознаването и прегледа на предметната област. Тук се засягат основните подходи и методи при решаването на проблеми свързани с реализацията на изследването и прилагането на методи за адаптация на игри от подобен ранг, спрямо кривата на учене. Прави се кратко описание на играта в помощ на изследването на тези методи.

Трета глава описва съществуващите технологии, анализира се необходимостта от използването на конкретни такива. За по-голяма точност технологиите се разделят по типове като се описва всяка една от тях, като се прави и сравнителен анализ.
Четвърта глава е описан анализа на изискванията. Описват се основните процеси и роли, документират се функционалните и не функционални изисквания. Детайлизира се необходимостта от външни системи за продукта.
Глава пета описва архитектурата на играта за стрелци от първо лице, създават се диаграми на поведението, декомпозиция на модули и под модули.
Глава шест разглежда конкретните методологии за подход по време не имплементацията. Описват се различните и нуждата от тяхното използване. Съпоставят се и се анализират първоначалните цели постиганите резултати. Разглеждат се начините за тестване на играта като се прави обобщение от резултатите от тестването на играта.
Седма глава – обобщение на свършената работа, обсъждат се идеите за нейното развитие

2. Преглед на предметната област на изследването на методите за адаптиране на видео игри спрямо кривата на учене на играча

2.1. Същността на методите за адаптация на видео игри

В днешно време видеоигрите са станали голяма част от нашата култура и ежедневие, и са едни от най-популярните форми за забавление. Те пристрастяват, вдъхновяват и държат вниманието на играча с часове. Игрите не са само за забавление, използват се от психолози, преподаватели, учени и маркетинг специалисти за да подобрят начина на учене, да привлекат вниманието, за анализиране поведението на мозъка. Такива игри които се използват не само за забавление, се наричат сериозни игри. Термин „сериозни игри“ е въведен от Sawyer и Rejeski.[8]

Чрез сериозните игри хората придобиват знания и умения, този тип игри имат позитивно влияние в областите обвързани с когнитивни, мотивационни, емоционални и социални проблеми. А повечето сериозни игри са от тип симулаторни игри, поради простата причина че те имат възможността максимално да представят реалния свят чрез виртуалният такъв, който предоставят.

2.1.1. Механизъм на работа

Методът, представен в дипломната работа за адаптиране е чрез крива на учене, наричана още крива на опита или крива на производителността [3] [4] [5] [6] [7], предоставя графично изображение на напредъка с който играчът се учи за даден период от време. В компютърните игри, кривата на учене илюстрира как играчът прекарва време си за овладяване на играта, чрез развиване на игровите си умения, познавателни способности и знания за разрешаване и справяне с предизвикателствата на играта. Кривите на учене биват различни, както за всяка игра, така и за различните играчи и представлява напредъкът в ученето и оттам специфичното представяне на отделния играч. Това означава, че е от голямо значение разпознаването на кривата на учене на играчът за реализацията на ефективен игров процес, защото това отразява мотивацията и ангажираността на играча и така тя може да се приложи за промяна на сложността на играта и нейните предизвикателства.

За лесното и автоматично разпознаване на кривите на учене при играчите като поведенчески шаблони за постиженията на играча, в игра от типа стрелба от първо лице (Box Shooter) ще се използва софтуерният компонент „Player-centric rule-and-pattern-based adaptation asset“ [4] [5] [9] създаден от екип на ФМИ в рамките на европейския проект RAGE (http://rageproject.eu/). Интеграцията му в играта ще помогне за динамично откриване на специфични криви на учене на играча представляващи цялостното му представяне за определено време. Практически експериментът включва разпознаване на характеристиките на шаблон в индивидуални криви на учене, след което се дефинират в компонента и се продължи с разпознаването им динамично по време на игра. [9]

Чрез внедряването на този компонент в игра стрелба от първо лице (Box Shooter) ще можем да анализираме и тестваме ползата от адаптации ориентирани към играча базирани на резултатите от играта. Играта трябва да отчита прогреса или регреса на уменията за каране на играча и спрямо тях да предприеме нужните мерки за да подобри техните постижения.

2.1.2. Функционалности

Видеоигри подобни на играта която е предмет на дипломна работа (игра за стрелци от първо лице) се характеризират със следните функционалности и механизми за работа:

- Графичното представяне на околната среда трябва да е максимално близо с тази в реалният свят. (реалистичен дъжд, мъгла, околния свят, оръжията с който се стреля, скоростта на обектите и количеството на различните типове обекти (в играта са заложени три типа обекти, при стрелба по първия обект се дават точки на играча, при стрелба по втория обект се увеличава времето за което трябва играчът да достигне определен Праг за да премине на следващо ниво и при стрелба по третия обект се отнема от времето) по който се прицелва и др.
- Визуализация на времето за което играчът трябва да постигне определен резултат.
- Визуализация на постигнатия резултат за предоставеното му време, тоест Прагът (точките от успешната стрелба) който е достигнал за времевия интервал
- Възможността да се играе както с онлайн десктоп версия, така и онлайн през браузър. Играчите ще имат възможност да играят от всякъде и по всяко време. Поддръжка на играта на различни платформи и операционни системи.
- Адаптиране на сложността на играта спрямо постигнатите резултати на играча. Адаптирането се състои в промяна характеристиките на околната среда, като промяна на видимостта чрез промяна на гъстотата на мъглата, чрез промяна гъстотата и силата на дъжда, чрез промяна на скоростта на обектите, промяна на размера на обектите по които играчът се прицелва.
- Записване на резултатите на сървър с цел изследване и по нататъшното им използване с цел подобряване резултатите на играчите като спрямо тези резултати ще могат да бъдат открити различните криви на учене.

2.1.3. Бизнес модел

Бизнес моделът на подобни игри от типа на стрелба от първо лице (Box Shooter) се базира не само върху нуждата от забавление или обучение на играча, но и на извличане на печалба от продаването на играта или предлагането на микро-транзакции с цел подобряването на положението или състоянието на играча, ако не иска да влага много усилия за да постигне целите на играта. Има много подобни игри и по-голяма част от тях, които са високо качествени, са платени, а тези които не са платени, не успяват да предоставят нужните качества на дадени аспекти, като реално и атрактивно представяне, инструменти и приспособления за измерване на различни характеристики, подходящи предизвикателства и др. С цел маркетинг този род видеоигри предлагат демо версия която е непълна но достатъчна за да предобият хората представа за това какво предлага играта, за да могат да си я закупят ако придобият интерес към нея

2.1.4 Характеристики и предимства

Игри от типа на сериозните игри в частност играта която е предмет на дипломна работа (стрелба по движещи се предмети от първо лице: Box Shooter) имат за цел не само забавление, а по – скоро усвояване на нови знания и умения. Употребата им може да допринесе много за сектори като образование, отбрана, здраве, наука. Могат да се използват както за обучение на пожарникари и военни, така и за обучения в областта на математиката или език. За правилната работа и за тяхната ефективност се гарантира чрез пет елемента включени в разработването им.

- Сюжет – повечето игри от този тип имат сюжет или основна история, колкото по-дълбоко е изградена историята толкова по-лесно се мотивират обучаемите
- Игровизация (gamification) – елемент на всяка игра е динамиката, която включва класиране, награди, значки или събиране на точки. Игровизацията мотивира обучаемите: всеки човек обича да печели монети или да преминава на следващо ниво. Класирането също е важно: конкуренцията със съучениците или колегите, окуражава хората да се справят по-добре
- Моментна и персонална обратна връзка – обученията на живо се осъществяват с още дузина души единствен преподавател, докато сериозните игри предлагат моментална и персонална обратна връзка. Обучаемият си взаимодейства директно с играта и получава награда или наказание веднага след действието, което е извършил. При по – сложните игри обратната връзка показва къде обучаемият е сбъркал, като се дава възможност той да се представи по – добре следващия път.
- Симулация – имитира се реалния свят, като се използват герои и се пресъздава сценарии, играчът се потапя в свят, който е много близък до реалния, като по този начин могат да се упражняват знания или умения в безопасна среда.
- Цел – основния елемент на игрите от този тип е обучението [10].

Когато горепосочените пет елемента бъдат успешно комбинирани е налице силен ефективен учебен инструмент, което от своя страна показва предимствата на този тип игра [10].

- Повишаване на ангажираността и мотивацията
- Насърчаване интелекта и желанието за размисъл по – важни теми
- Предоставят близък до реалността, но безопасен терен за практика
- Подобряват усвояването и запомнянето.
- Окуражават обучаемите да проявяват самодисциплина и контрол
- Осигуряват мониторинг на обучителния процес
- Насърчават креативността и меките умения
- Дават възможност на обучаемите да подобрят компютърните си умения [10].

Една от предимствата на игри от тип стрелба от първо лице е опростения дизайн, няма анимиран аватар, няма AI (Artificial Intelligence) който да контролира камерата лесна итерация и позициониране.

2.1.5 Проблеми и недостатъци

Съществуват редица проблеми по време на разработката и след това на игри от подобен характер, като тези проблеми могат и да се превърнат в недостатъци на играта.

- Разработването на дизайна на виртуална среда която е максимално близко с реалния свят и в която е поставен героя. Това включва имплементирането на реални звукови и визуални ефекти (нереалистична околна среда, нереалистична мъгла, липсата на звук от дъжда, трудно различимите обекти на екрана поради лоши дизайнерски решения, изместване на мерника което води до неточност и др.).
- Разработването на реално поведение на героя във виртуалната среда.
- Разработването на AI (Artificial Intelligence) за NPC (Non-player character - играчи които се управляват от алгоритми и не са под контрола на главния играч) може да бъде прекалено сложно за изпълнение (при големи битки в тактическите игри със стрелба) или може да бъде прекалено наивен което пък от своя страна води до загуба на ефективност и интерес.
- Системата която контролира поведението на въображаемата камера се нарича модел на камерата (camera model). Статичния модел винаги показва виртуалното пространство от фиксирана перспектива, докато динамичния модел движи въображаемата камера в зависимост от действията на героя или събития предизвикани от него, изисква повече усилия за проектиране и изпълнение, но въпреки това превръщат играта в по-кинематографичен. Относно игри за стрелба от първо лице (first-person shooter) графиката винаги е 3D и камерата заема позицията на аватара (с други думи от позицията на очите на героя). Играча обикновено не може да види тялото на аватара си, въпреки че в някои игри се вижда само ръцете на героя което води до по – малко удоволствие от играта (по-малко потапяне в играта тъй като не вижда езика на тялото, изображението на лицето). Драматичните ефекти липсват, понеже няма възможност за използване на кинематографични ъгли на камерата.
- Събирането и анализирането на коректни данни от проведени практически тестове.
- Разработването на една такава игра като стрелба от първо лице (напр. FPS Box Shooter) може да бъде много скъпо не само спрямо използваният софтуер но и необходимия хардуер.
- Обемът на изследванията, проведени в проблемната област, е недостатъчен, което води до недостатъчна теоретична база и ограничения при синтеза на модели на обучаващи игри, дидактически модели и игрово ориентирани модели на обучавания. Изследванията в проблемната област са свързани и с мобилизирането на определени ресурси (човешки, апаратни, софтуерни, финансиране, времеви и др.)

2.2. Съществуващи решения

В процеса на разработка на тази дипломна работа ще се разгледат и сравнят различни игри от типа на FPS (стрелба от първо лице).

2.1.1. Red Alliance

Игра от типа FSP (first-shooter player), в чиято атмосфера на тъмни, екшън трилъри с елементи на ужас, играчът може да се потопи. Действието се развива през 21-ви век в източна Европа където властва репресивен режим. Страната в която попада главния герой е обвита от лъжи разпространени от загадъчен доктор с името Грей, амбициозен учен опитващ се да постигне контрол над ума чрез нелегални изследвания. Целта на главния герой е да достигне бунтовническа група „Червения Алианс“, която е все още против репресивния режим – и да сложи край на лудостта на д-р Грей. Играта се отличава с класическото стрелба от първо лице като предоставя и втори режим на игра - Arcade. На играча се предоставя възможност за опционални секретни секции като се позволява на играча сам да избира собствения си подход и стил на игра. Кампанията се състой от 9 игрални глави, всяка от които се провежда в различна обстановка. Със своя напредък в играта, атмосферата се променя. Дава се възможност на главния герой да участва в случайни събития, които да го възнаградят след завършването им. [11]

Фиг. 1 – Играта Red Alliance

2.1.2. Fallout 4

Bethesda Game Studios, спечелила награди за създаването на Fallout 3 и The Elder Scrolls V посрещат играча в света на Fallout 4, която е и най-амбициозната им игра. Като единствен оцелял героя се потапя в свят унищожен от апокалиптична ядрена война. Играта дава свободата на играча да прави каквото пожелае в един масивен открит свят със стотици местоположения, герои (контролирани от AI на играта) и т. нар. мисии или куестове (quests). Куестът или мисията е задача във видеоигрите, при която играчът контролира герой или група от герои с определена крайна цел, за да получи награда. Мисиите са най-често срещани в ролеви игри и масова мултиплеър онлайн игри. ["Effective Quest Design in MMORPG Environment". Game Developers Conference 2005, March 11, 2005]
Играчът може да избере аватар по свой избор чрез вградената система в играта – S.P.E.C.I.A.L определяща характера на героя. От силен брониран войник до харизматичен гладък говорещ човек. Играта предлага изцяло нови поколение графични и осветителни тела в света на Fallout, графиката е изпълнена с динамични детайли. Вградената система V.A.T.S (динамична система за подпомагане на насочването) позволява на играча да избира атаките си наслаждавайки се като на кино. Оръжия, броня, химикали и храна са само началото – дори играчът може да изгради и управлява цели селища. [12]

Фиг. 2 – Играта Fallout 4

2.1.3. Left 4 Dead 2

Играта е комбинация от Action и Horror и е FPS игра (first-shooter player) която потапя играча в градове, блата и гробища на дълбокия юг, от Савана до Ню Орлианс в пет кампании. Героят е един от четирите оцелели въоръжен с широк спектър от опустошителни класически и модернизирани оръжия, в допълнение на това се дава възможност на играчът да излее агресията си върху заразените контролирани от вградения AI играчи с помощта на оръжия от верижни триони до така смъртоносни тиган. Тази игра е от следващо поколение игри в сътрудничество с производители на игри като Half-Life, Portal Team Fortress и Counter-Strike. Аватара на играча е снабден с над 20 нови оръжия за близък бой като: брадва, резачка, тиган, бейзболна бухалка и т.н. Играта разполага с история, и предоставя на играча диалог с останалите оцелели. Може да се играе в пет експанзивни кампании в режим на игра Versus и Survival. Разполага с мултиплеър (с множество играчи) режим. Разширена технология на AI Director 2.0 наречена „The AI Director“ предоставя уникален геймплей (gameplay) на Life 4 Dead 2 – персонализиране на вражеското население, ефекти и музика въз основа на представянето на играчите, персонализира оформлението на ниво обекти, осветление, време (за да отразява различните часове на деня). [13]

Фиг. 3 – Играта Left 4 Dead 2

2.2.4. Doom

ID Software, студио което е пионер в жанра FPS (first-person shooter) която е и създател и на мултиплеър играта Deathmath се завръща с играта Doom която е брутално забавно и предизвикателно преживяване за играча. Безмилостни демони, невъзможно разрушителни оръжия, бързо или плавно движещи се обекти дават възможност на играча да се наслади на битки със стрелба от първо лице. Историята на играта поставя аватара на играча в центъра на изследователско съоръжение на Съюза на аерокосмическата корпорация на Марс, която е затрупана от свирепи и мощни демони. Само един човек стой м/у техния и нашия свят. На играчът се дава възможност да комбинира арсенала си с футуристични и емблематични оръжия, ъпгрейди, система за близък бой.
Играта разполага с технология Doom SnapMap – мощен инструмент за редактиране на нивата, който позволява неограничен геймплей преживяване на всяка платформа. Без предишен опит в играта всеки играч може бързо и лесно да комбинира и визуално да персонализира картите, да добавя предварително дефинирани или напълно персонализирани геймплей и дори да редактира логиката на играта, за да създава нови режими на игра. [14]

Фиг. 4 – Играта Doom

2.2.5. Sniper Ghost Warrior 3

Sniper Ghost Warrior 3 разказва история за братство, вяра и предателство давайки на героя най-добрия опит в боравенето със снайпер. Ролята на играча се поема от американски снайперист на име Джонатан Север, който е пуснат на вражеска територия. Играчът има възможност да разглежда карти в динамично време на дневен и нощен цикъл и на базата на решенията които главния герой е взел се променя и играта. Главния герой разполага с персонализирано оръжейно оборудване, аксесоари, превозни средства, дрон – устройства, освен това той има възможност да избере пътя по който да постигне крайната цел на играта. Изкуствения интелект заложен в играта контролира целите и обхвата, скоростта и посоката на вятъра, контрола на дишането което прави играта много по реалистична. В самата игра е заложен stealth gameplay режим което кара играча да се чувства и да изживява сцените и препятствията по пътя до крайната цел на играта като истински снайперист. Изборът на оръжия е съобразен с мисиите които са възложени във всяко едно ниво както и на личния стил на игра на играча: пушки, картечници, експлозиви и много други такива. [15]

Фиг. 5 – Играта Sniper Ghost Warrior 3

2.3. Сравнителен анализ на съществуващите решения

Съществуващите решения ще сравним като използваме следните критерии за оценка:

- Цена – Един от най-важните критерии, тъй като се отнася до всички потребители. Повечето симулатори имат демо версии които могат да се тестват за да преценят хората дали да си закупят пълната версия.
- Поддръжка на различни платформи. Може ли играта да се игра на операционни системи като Windows, Mac, Unix, Android, iOS или на конзолна платформа (като например PS, Xbox, Nintendo и др.).
- Адаптивност на дизайна. Дали играта поддържа работа с различни по размер на екрана, устройства (поддръжка на различни резолюции).
- Играта с отворен код ли е? Това означава дали е дадена възможността на потребителите да пригодят и модифицират играта според собствените си нужди или да правят подобрения които ще са общодостъпни за всички за да могат да бъдат използвани.
- Поддържани режими на играта (**сингъл-плейър, мулти-плейър**).
- Системни изисквания. Какви са минималните и максималните системни изисквания за играта, за да върви на платформите на които се поддържа.

Нужда от специален хардуер. Повечето симулатори изискват специални устройства (джойстик, контролер и др.) за да могат да бъдат използвани, възможно е и само с клавиатура и мишка да се играе.

	Red Alliance	**Fallout 4**	**Left 4 Dead 2**	**Doom**	**Sniper Ghost Warrior 3**
Цена	9.99 €	14.99 €	8.19 €	19.99 €	19.99 €
Демо Версия	No	No	No	No	No
ОС	Windows 10, Windows 8.1, Windows 8, Windows 7 Service Pack 1, Windows Vista Service Pack 2	Windows 7, Windows 8, Windows 10 (64 bit OS required)	Windows 7 (32/64 bit), Windows Vista (32/64 bit), Windows XP, MacOS 10.7, Ubuntu 12.04	Windows 7, Windows 8.1, Windows 10 (64 bit OS required)	Windows 7, Windows 8.1, Windows 10 (64 bit OS required)
Адаптивност на дизайн	Yes	Yes	Yes	Yes	Yes
Open Source	No	No	Yes	No	No
Single Player	Yes	Yes	Yes	Yes	Yes
Multi Player	No	No	Yes	Yes	Yes
Минимални системни изисквания	Processor: Intel Core i5 3470 @ 3.2 GHz or AMD X8 FX-8350 @ 4 GHz Memory: 6 GB RAM	Processor: Intel Core i7 4790 3.6 GHz / AMD FX – 9590 4.7 GHz Memory: 8 GB RAM	Processor: Intel Core 2 duo 2.4 GHz Memory: 2 GB RAM Graphics: Video Card Shader model	Processor: Intel Core i5 – 2400 / AMD FX-8320 or better Memory: 8 GB Graphics: NVIDIA GTX	Processor: Intel Core i3 3240 3.4 GHz or AMD FX – 6350 3.9 GHz Memory: 8 GB Graphics: NVIDIA GeForce GTX 660 2

	Graphics: NVIDIA GTX 660 2GB/ AMD HD 7870 2GB DirectX: Version 11 Storage: 5 GB	Graphics: NVIDIA GTX 780 3GB/ AMD Radeon R9 290X 4 GB Storage: 30 GB Additional Notes: Full gaming controller support	3.0 NVidia 760o, ATI X1600 or better DirectX: Version 9.0 c Storage: 13 GB Additional Notes: Broadband internet connection for multiplayer. Full gaming controller support	670 2 GB / AMD Radeon HD 7870 2GB or better Storage: 55 GB Additional Notes: Requires Steam activation and broadband internet connection for multiplayer. Partial gaming controller support.	GB or AMD Radeon HD 7850 2GB DirectX: Version 11 Storage: 50 GB Additional Notes: Online Connection Requirements 512 KBPs or faster internet connection. Partial gaming controller support.

Таблица. 1 – Сравнителен анализ на съществуващи решения за игри от тип FPS (игри за стрелба от първо лице)

От така направеният анализ, представен на „Таблица 1“ може да се установи, че всички разгледани игри разполагат със сходни характеристики по отношение на поддръжката на различните операционни системи, адаптивността на дизайна към различните размери на екранните устройства. Сходство има също така в системните изисквания които се стремят да се използва най-новия хардуер съществуващ на пазара. Друго сходство е интегрирането на различни аксесоари даващи възможността да се играе по различни начини, удобни за играча било то с мишка, клавиатура или игров контролер. Но нито един от тези игри не използва методите за динамично адаптиране на сложността спрямо резултатите на играча, което е и предмет на разработка на дипломната работа. Ето защо е нужно провеждането на изследването на тези методи и прилагането им върху FPS игра разработена по време на дипломната разработка.

2.4. Изводи

От направения преглед на съществуващи решения става ясно, че при разработването на игри от FPS (First Person Shooter, игри за стрелба от първо лице) жанра трябва да се наблегне на някои функционалности и механизми: играта да поддържа всички възможни операционни системи, да предлага възможността да се играе с различни резолюции, да предоставя възможност да се играе, както на по нови системи, така и на по стари такива. Също така да се интегрират методите за адаптиране на играта, както и възможността да се играе чрез различни устройства. От голямо значение ще е играта да е общодостъпна за всички без да се заплаща за нейното използване.

3. Избор на технологии и платформи за разработка

3.1. Изисквания към средствата за реализация и предварителна оценка

Анализът и оценката на функционалните и нефункционалните изисквания за игри от тип FPS (first-person shooter) показват необходимостта от използването на технологии които ще позволят разработване на реалистична виртуална околна среда, както и реализъм спрямо доста аспекти свързани с играта като реалистична околна среда, реалистична мъгла, звук от падащия дъжда, движението на обектите и звука от удара на куршума с обекта. Друг фактор на който трябва да се наблегне е изборът на подходящи технологии позволяващи играта да се играе на различни операционни системи и платформи, както и поддръжката на различни резолюции както за стари системи така и за по нови такива. Потребителският интерфейс трябва да е интуитивен и възможно най-много опростен. Разбира се, едни от водещите изисквания за набор от технологии са бързодействието, надеждността, осигуряването на бърза разработка и лесна разширяемост. Не трябва да се забравя че за избора на технологии е необходимо да се помисли са възможността за бързо изпращане на резултатите от играта докато играчът играе към подходящ сървър където да бъдат обработени и записвани. Критерий за избор на технологии е методите за адаптиране на сложността на играта. При избора е важна и теологията което искаме да използваме и по-точно нейната популярност която да гарантира бъдеща поддръжка на играта и лесна и интеграция с други популярни технологии и компоненти.

Предварителен анализ на технологиите и тяхната евентуална реализация включва следните под категории:

- Платформа за разработка на игри – основната задача е да се избере подходяща платформа за имплементиране на игра от тип FPS (First Person Shooter) и различните компоненти които я изграждат. Важна характеристика по която да се избере платформа ще е предоставянето на готови компоненти които ще могат бързо и лесно да се имплементират и интегрират. Основен фактор са езиците за програмиране които поддържа платформата. И не на последно място, дали платформата поддържа технология позволяваща направените игри и приложения на нея да се използват на различни операционни системи.
- Технология за изпращане на данни от играта до даден сървър – поради факта, че трябва постоянно да се изпращат резултатите по време на игровия процес, трябва да се избере технология която да поддържа връзка към даден сървър, към който да се изпращат и записват данните за всеки един играч, в отделни файлове.

Методи за адаптиране трудността на нивата в играта– както се установи по време на анализа на съществуващите решения в разглежданата проблемна област, ще има различни методи за адаптиране. Поради тази причина ще се направят различни версии на играта, като всяка версия ще използва различен метод за адаптация. Това ще е и единствената разлика между версиите, всичко останало като имплементация ще е едно и също. Целта е да можем да тестваме всяка една версия и след това въз основа на резултатите от тестовете да преценим кой от методите е най-удачен за промяната на сложността на играта.

3.2. Избор на платформа за разработка

В анализа който направихме се вижда че избора на платформа за разработка на такъв тип игри, се свежда и до това кой програмен език ще се използва и дали избрания език е ефективен за нуждите на разработката. Изборът на програмен език е продиктуван вече съществуващи познания, както и наличието на предишен опит в използването им. Използваният език ще бъде C# - това е обектно-ориентиран език за програмиране, разработен от Microsoft, като част от .Net, с общо предназначение. За по голямо улеснение при разработката на играта трябва да се направи избор за конкретна платформа която го използва.

3.2.1. Налични платформи

Constructor (Scirra, Studio 414 The LightBulb**)** – Constructor е платформа за разработване на уеб базирани приложения. Използва HTML5 и JavaScript като езици за програмиране. HTML5 е сравнимо със Adobe's Flash технологията която доминира в интернет игрите. За разлика от Flash обаче не е нужно да се инсталира допълнителен софтуер, за да се играе игра направена на HTML5 тъй като е вграден в браузъра. Поради тази причина игрите направени на Constructor могат да вървят и на мобилни устройства. JavaScript се използва за подобряване бързината на играта и за писане на допълнения (plugins) към играта които могат да бъдат използвани и от други хора. Също така Constructor предлага offline режим на играене използвайки HTML5 AppCache, позволявайки на игрите да се играят дори когато няма достъп до интернет. [16]

GameMaker: Studio (YoYo Games 2007) - GameMaker: Studio е платформа за разработване както на 2D уеб базирани игри, така и на 2D десктоп и конзолни игри. Платформата използва вграден език за програмиране GML, както и 'Drag and Drop' програмиране. Платформата предлага внедряване на играта на различни операционни системи, конзоли и платформи (Windows, Mac, Ubuntu, Android, iOS, HTML5, PlayStation 4, Xbox one и др.). Предлага голям набор от помощни инструменти за персонализиране на почти всички аспекти в играта. Също така дава възможност за разработване на собствени модули както и използвани и на чужди такива. [17]

Unity (Unity Technologies) – Unity е платформа за разработка на 2D и 3D игри, които могат да бъдат уеб базирани, десктоп или конзолни. Основни езици които са интегрирани в платформата и могат да бъдат използвани за програмиране на игрите са C# и UnityScript познат още като JavaScript for Unity и Boo. Освен че предлага внедряване на игрите за почти всички операционни системи и платформи, също така предлага набор от мощен инструментариум за създаване на двуизмерни и триизмерни обекти, които да бъдат използвани в играта, за разработване на поведението на тези обекти, за разработване на отлични графични среди. Има съвместимост със сорс контрол технологии които помагат за управлението на разработката на играта, лесното и споделяне между хора работещи колективно за създаването на дадена игра. Възможност за програмиране на игри за виртуална реалност с новият вграден Vuforia Engine 7.5. Също така компанията е направила магазин за добавки (plugins/assets), достъпни за всички използващи тази платформа. Други технологии които се предлагат е възможността игрите да са мулти-плейър, а също и сингъл-плейър. [18]

Unreal Engine (Epic Games, Inc. 2004) - Unreal Engine е платформа за разработка на 2D и 3D игри, които могат да бъдат уеб базирани, десктоп или конзолни. Езиците вградени в тази платформа са C++ и BluePrints Visual Scripting. Това позволява максимално представяне на игрите направени на тази платформа. Поддържа внедряване на почти всички операционни системи и платформи. Също така предлага инструменти за разработка и модифициране на графични обекти. [19]

3.2.2. Сравнителен анализ

Сравнителния анализ има за цел да даде подходяща аргументация при избора на платформа. За да бъде възможна аргументацията, първо трябва да се заложат основните критерии, от които ще се определи използваната платформа. Анализът ще бъде направен по следните критерии:

- Цена за използване на платформата.
- Възможност за безплатно използване.
- Вградени програмни езици в платформата.
- Тип игри които могат да се разработят чрез тази платформа (2D, 3D)
- Поддържани платформи за внедряване (publishing a game).
- Възможност за разработване на собствени модули.
- Съвместимост с външни тулове.
- Системни изисквания.

	Constructor	**Game Market Studio**	**Unity**	**Unreal Engine**
Цена	Минимална цена: 89.99 €	Минимална цена: 39.99 €	Минимална цена: 30.00 €	Безплатен за използване. 5% ако разработената игр има печалба от 3000$ на четиримесичие
Безплтна версия	Временно ползване	Временно ползване	да	да
Вградени езици	HTML5, JavaScript	GML	C#, UnityScrip (also known as JavaScript for Unity), Boo	C++
Типове поддържани игри	2D, Уеб базирани игри	2D, Отчасти 3D, Онлайн и офлайн десктоп игри	2D,3D, уеб базирани, дестоп игри конзолни игри	2D,3D, уеб базирани, дестоп игри конзолни игри
Платформи за внедряване (publishing)	Windows, Mac, Linux, iOS, Android,	Windows, Mac, Ubuntu, PS 4, Xbox One, iOS, Android, Windows phone, Tizen	Windows, Mac, Ubuntu, Linux, PS 4, Xbox One,	Windows, Mac, Ubuntu, Linux, PS 4, Xbox One, iOS, Steam, HTML5,

	Windows phone, Wii U		iOS, Android, Facebook	Nintendo switch, PlayStation VR, Oculus Rift VR, VIVEPORT VR, Daydream
Разработване на собствени модули	Да	Да	Да	Да
Съвместимост с външни тулове	Да	Да	Да	Да
Single Player	Да	Да	Да	Да
Multi Player	Да	Да	Да	Да
Системни изисквания	CPU: Quad Core CPU or Dual Core CPU (Intel Core 2.8 GHz, AMD Athlon 64 X2 4400+ or faster) CPU SPEED: Info RAM: 8 GB OS: Windows 7, 8, 10 VIDEO CARD: DirectX 9 Compatible - Nvidia Geforce 8800GT / ATI Radeon 4850 or faster with Shader Model 3 and 512 MB VRAM FREE DISK SPACE: 10 GB	CPU: Info CPU SPEED: 64bit Intel compatible Quad Core CPU RAM: 8 GB OS: Microsoft 64bit Windows 10 VIDEO CARD: DX11 based graphics card FREE DISK SPACE: 3 GB	CPU: SSE2 instruction set support. GPU: Graphics card with DX10 (shader model 4.0) capabilities. The rest mostly depends on the complexity of your projects. Additional platform development requirements: iOS: Mac computer running minimum macOS 10.12.6 and Xcode 9.0 or higher. Android: Android SDK and Java Development Kit (JDK); IL2CPP scripting backend requires Android NDK.	Processor:Quad-core Intel or MD, 2.5 GHz or faster Memory: 8 GB RAM Video Card/DirectX Version DirectX 11 compatible graphics card

			Universal Windows Platform: Windows 10 (64-bit), Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK	

Таблица. 2 – Сравнителен анализ на съществуващите платформи за разработване на игри

3.2.3. Заключение и обосновка

В заключение може да се каже че Unreal Engine е най-подходяща платформа за разработването на игри, притежава добре описана документация в помощ на разработчиците на игри под формата на видеа за обучение и упражнение. Освен това е безплатна, при условие че вече разработената игра печели на четиримесечие повече от 3000$ то в този случай компанията Epic Games разработила платформата взима 5% от печалбата на играта, и не на последно място предлага повече платформи за внедряване на играта. Но има един съществен недостатък за целите на дипломната работа а това е че поддържа само език за програмиране C++, тъй като играта която е предмет на дипломната работа трябва да бъде разработена на език за програмиране C#, то остава като избор за платформа Unity. Някои от недостатъците на тази платформа спрямо Unreal Engine са поддържаните платформи за внедряване на играта и ограничената безплатна версия .която се предлага от Unity. Като изключим тези недостатъци, то тази платформа притежава всичко необходимо, за да бъде предпочитания избор.

3.3. Избор на технология за изпращане на данните от играта до даден сървър

За да е достъпна играта до по голям кръг от хора е необходимо играта да се играе през интернет, което води до въпроса как данните на всеки играч ще се събират за анализ. Отговора на този въпрос се крие в наличните технологии които предоставят различните платформи за разработки на игри. Това което разбрахме до сега е че Unreal Engine е най-подходящ от гледна точка на това че е безплатен и мощен инструмент за създаването на такава игра, проблема е в езика който стой в основата на разработката. Поради тази причина нашия избор се спря на Unity платформата, която е подходяща за нашите цели.
Unity платформата разполага с налични технологични решения за изпращане на данни от играта до даден сървър.

3.3.1. Налична технологии

Именувани тръби: Named pipes е разширение на концепцията за традиционните pipes в Unix и Unix – подобни системи и е един от методите за комуникация между процесите (IPC). Тази концепция се среща и в OS/2 и Microsoft Windows. Използват се еднопосочна и дуплексна комуникация между един сървър и един или повече клиенти. Всички инстанции на named pipes споделят едно и също име но всяка инстанция има свой собствен буфер като предоставя отделен канал за комуникация клиент – сървър.
Тази технология обикновено се използва заедно с нишковото програмиране. Особени характеристики на тази технология е че трябва да се разработи и поддържа клиентска и сървърна част които да си комуникират. [20]
Sockets (комуникират чрез TCP/IP**) –** Осигуряват лесен начин за комуникация по мрежата с цел обмяна на данни. Комуникацията чрез сокети се базира на архитектурата клиент – сървър. Сървъра предоставя порт и мрежови адрес на който изчаква клиента да се свърже. Когато клиента се свърже със сървъра използвайки сървърния порт и адрес започва обмяна на данни, като тази обмяна представлява размяна на специализирани данни и съответно тяхната десериализация. Когато обмяната на данни приключи, връзката се прекъсва. Благодарение на мощните си функции сокетите се превръщат в необходимост от изучаването им от страна на разработчиците. [21]
Уеб услуги (Web Service) – Тази технология разчита на това, че даден сървър ще изпращат данните под формата на заявка, която ще бъде обработена от уеб услугата. Уеб услугата представлява код който се изпълнява от страна на сървъра. Единственото изискване към клиента е да направи коректна заявка към уеб услугата намираща се на дадения сървър. [22]

3.3.2. Сравнителен анализ

Обективната преценка и аргументация относно избор на технология за изпращане на данни от играта до даден сървър се базира на следните критерии:

- Компоненти за имплементиране
- Използване допълнителни технологии
- Възможни ефекти от използвани външни технологии

	Named Pipes	Sockets	Web Serices
Компонент за имплементиране	Клиентска част която да изпраща съобщения, Сървърна част която да приема съобщения.	Клиентска част която да изпраща съобщения, Сървърна част която да приема съобщения.	Сървърна част която обработва съобщенията, Клиентска част изпращаща съобщения.
Използване на допълнителни технологии	Нишки (Threads)	Сериализация на данни	код съдържащ логиката за обработка на данни
Възможни ефекти от използвани външни технологии	Може да се получи deadlock, ситуация при която ще се забави играта	Сериализацията и десериализацията забавя процеса на изпращане на данните	Ако сървъра падне, уеб услугата няма да е достъпна, но това няма наруши работа

			на играта и съответно няма да я забави.

Таблица. 3 – Сравнителен анализ на съществуващите технологии за пращане на данни от играта към даден сървър

3.3.3. Заключение и обосновка

На пръв поглед Named Pipes и Sockets технологиите не изглеждат добро решение за изпращане и получаване на данни между играта и сървъра, тъй като има възможност да прекъсне или забави процеса на играта. Това което се вижда е че са по трудни за имплементиране и интегриране от колкото използването на уеб услуги.
Уеб услугите като технология не представляват проблем за играта дори и да не работят. На базата на направеният анализ и различните разгледани критерии, уеб услугите са предпочитаната технология за използване комуникация между играта и сървъра.

3.3.4. Избор на методи за адаптиране на сложността

Методите за адаптиране на сложност при вече разгледаните платформи за разработка се определят от различни критерии, фактори и ограничения така както се определят и методи за комуникация между сървър и клиент, така както се прави избор на платформа за разработка на играта. Тези методи на адаптация не трябва да забавят игровия процес, трябва да е лесен за интегриране. Освен всичко това адаптивните методи които подлежат на избор за различните версии на играта трябва да отговарят на условия за провеждане на изследване за адаптация на играта спрямо кривата на учене.

3.3.5. Налични технологии

Първи метод – промяна на трудността в началото на играта. Преди да започне играта играчът може да настрои трудността на играта на която би искал да играе, спрямо която се настройват различните характеристики определящи трудността. Повечето компании разработващи игри използват този метод за адаптиране сложността на играта. По този начин трудността на играта е една и съща през цялото време докато не бъде сменена от самия играч.

Втори метод – Без промяна на трудността. В някои игри е заложена стратегията, всяко следващо ниво да е с по-голяма трудност спрямо предишното. В този случай играчът няма възможност да променя трудността на играта.

Трети метод – Промяна на трудността зависи от показателите на играча по време на игровия процес. Спрямо тези показатели по зададени нива, в случая резултата който може да достигне играчът, ще се определя каква адаптация ще се приложи в играта.

Четвърти метод – Промяната сложността се извършва по следният начин: първо се открива крива на учене, описана от метриките за резултата и разпозната динамично (в реално време) чрез анализ на шаблони. След което спрямо тази крива на учене ще се

извърши адаптация на динамичната сложност на играта по подходящо избрани характеристики.

3.3.6. Сравнителен анализ

Изборът на подходящите адаптивни методи за играта която е предмет на дипломна работа ще се прави въз основа на няколко критерии: начинът предлаган от методите за адаптиране на сложност (начин на адаптиране), ограниченията поставени от разработката на дипломната работа и накрая от избраните предишни технологии за избор на платформа и метод за комуникация между играта и сървъра.

	Метод 1	**Метод 2**	**Метод 3**	**Метод 4**
Начин на адаптиране	В самото начало на играта (статична адаптация)	Не се прави адаптация на сложността (вградена е в играта и не може да се променя)	Динамична адаптация на сложността по време на игровия процес спрямо резултатите на играча	Динамична адаптация на сложността по време на игровия процес спрямо резултатите на играча
Статично адаптиране на сложността	Да	Да	Не	Не
Отговаря ли на ограничения от страна на дипломната работа	Не	Да	Да	Да

Таблица. 4 – Сравнителен анализ на съществуващите методи за адаптация на сложността на играта

3.3.7. Заключение и обосновка

От „Таблица. 4“ и от направения анализ на отделните решения за адаптация на сложността на играта се вижда, че метод 1 не е подходящ за целите на дипломната работа поради факта че следенето на резултатите на играча както и анализирането им е почти невъзможно. Метод 2 е подходящ за нашите цели тъй като този метод не предлага адаптация на сложността, ще го използваме като базова основа за сравнение спрямо останалите модели, чрез този метод ще се съберат данни с помощта на който да сравним резултатите и то другите методи и да се направи оценка за подходяща адаптация на сложността на една видеоигра. Останалите методи (метод 3 и метод 4) за напълно подходящи методи за динамично адаптиране.

4. Анализ на изследване на методите за адаптиране на игри спрямо кривата на учене

Анализирането на изискванията е един от най-важните етапи от разработката на софтуер. Това е допирната точка между бизнес логиката и технология.

4.1. Основни концепции

В текущото изследване това са методите за адаптация на игри за стрелци от първо лице, тяхното приложение и разработване. Тези методи е необходимо да предлагат динамично адаптиране на сложността (промяната трябва да се извършва в реално време). Изключение прави базовия модел който ще се използва като основа за сравнение с останалите методи, като при него няма нужда да се реализира адаптация на сложността.
Методите трябва да предлагат интуитивен начин за анализиране на резултата на играча според който ще се адаптира трудността на играта.

4.2. Функционални изисквания

В играта за стрелци от първо лице ще бъдат имплементирани три варианта за адаптация на сложността което от своя страна води до три версии на играта, като всяка една от версиите ще представлява различен метод за адаптация. Поради тази причина ще се обърне внимание на изискванията за всяка от версиите.

4.2.1. Изисквания към играта – стрелба от първо лице

Неадаптивна версия

- Играчът се запознава с правилата на играта като преди да стартира игровия процес трябва да въведе e-mail за идентификация, целта е играчът да се идентифицира с формата от въпроси с която е съпроводена играта с цел настоящето проучване
- Следене на времето за което играчът трябва да достигне максимален праг за всяко ниво в отредения му времеви прозорец за всяко ниво
- Проследяване на скоростта с която се движат обектите (targets), посоката в която се движат, големината и гъстотата на обектите около играча.
- Отчитане на текущия праг.
- Използват се аудио и визуални ефекти в играта
- Добавяне на допълнителни ефекти (промяна на заобикалящата го среда) на всяко следващо ниво на играта
- Прицелването на обектите ще се осъществява посредством мерник на екрана задвижван от мишката.
- Резултатите на играча се обработват и записват в реално време.

Адаптивна версия спрямо зададени нива

- Играчът се запознава с правилата на играта като преди да стартира игровия процес трябва да въведе e-mail за идентификация, целта е играчът да се идентифицира с формата от въпроси с която е съпроводена играта с цел настоящето проучване
- Следене на времето за което играчът трябва да достигне максимален праг за всяко ниво в отредения му времеви прозорец за всяко ниво
- Проследяване на скоростта с която се движат обектите (targets), посоката в която се движат, големината и гъстотата на обектите около играча.
- Отчитане на текущия праг.
- Използват се аудио и визуални ефекти в играта
- Добавяне на допълнителни ефекти (промяна на заобикалящата го среда) на всяко следващо ниво на играта
- Прицелването на обектите ще се осъществява посредством мерник на екрана задвижван от мишката.
- Резултатите на играча се обработват и записват в реално време.
- Адаптиране на сложността на играта в реално време спрямо постигнатите резултати на играча.
- Достигайки определени предварително зададени нива на отчетените резултати на играча ще се стартира адаптация на сложността която ще се засилва или ще се намалява.

Адаптивна версия спрямо криви на учене

- Играчът се запознава с правилата на играта като преди да стартира игровия процес трябва да въведе e-mail за идентификация, целта е играчът да се идентифицира с формата от въпроси с която е съпроводена играта с цел настоящето проучване
- Следене на времето за което играчът трябва да достигне максимален праг за всяко ниво в отредения му времеви прозорец за всяко ниво
- Проследяване на скоростта с която се движат обектите (targets), посоката в която се движат, големината и гъстотата на обектите около играча.
- Отчитане на текущия праг.
- Използват се аудио и визуални ефекти в играта
- Добавяне на допълнителни ефекти (промяна на заобикалящата го среда) на всяко следващо ниво на играта
- Прицелването на обектите ще се осъществява посредством мерник на екрана, задвижван от мишката.
- Резултатите на играча се обработват и записват в реално време.
- Адаптиране на сложността на играта в реално време спрямо постигнатите резултати на играча.
- Адаптирането ще се осъществява според отчетените текущи резултати, като по този начин ще се открива кривата на учене, по която да се определи промяната на сложността

4.2.2. Идентифициране на отделни роли в системата

В играта стрелец от първо лице (FPS – First Person Shooter) можем да дефинираме следните роли:

GR1. Играч – човек ползващ аватар с помощта на който изпълнява условията на играта, действията на аватара (герой) се управляват чре команди задавани с помощта на входно устройства.

GR2. I/O Devices (like, Mouse, keyboard, speaker, monitor) – Устройствата, които се използват за въвеждане на данни, се наричат входни устройства (Input Devices). Входното устройство може да чете данни и да ги преобразува във формат които компютъра разбира и може да използва за подходящи цели. Изходните устройства генерират готовия продукт от машинната обработка във форма която хората разбират. Периферните устройства като мишка, клавиатура, монитор, джойстици, наричани още входно-изходни устройства (I/O devices) играят основна рола в това една игра да е интерактивна. Играчът въздейства на играта посредством входните устройства докато играта връща резултат от въздействието посредством изходните устройства.

GR3. Graphics Engine – или графичен енджин (двигател) е софтуер с помощта на приложен програмен продукт е способен да начертае графиките и графичните изображения върху изходно устройство, в частност дисплея (монитор) на компютъра. Думата "енджин" (думата „двигател“ не е прието да се използва) в компютърната област се отнася до софтуер, който помага за извършването на определен тип обработка на програми, като например енджин за текстова реч, енджин за бази данни, енджин за оформление или графичен енджин. Графичният енджин помага за подобряване на графиката на играта чрез увеличаване на резолюцията и увеличаване на броя на пикселите на единица площ. Този енджин прави сцените на играта да са ясни и да работят гладко.

GR4. Sound/Audio Engine - Аудио / звуковият енджин е компонентът, който се състои от алгоритми и е отговорен за обработването на звука. Вградените програми се записват в него, за да се справят с вградените в играта звукови ефекти. Той има способността да извършва изчисления с помощта на CPU или на всяка ASIC (Application Specific Integrated Circuit).

GR5. Rendering and Vision-Input Engine – Енджин за рендиране заедно с входната система за видения (зрение) произвежда 3D анимирани графики, използвайки различни техники, като растеризация и проследяване на лъчи. Те биват програмируеми или компилирани, за да бъдат изпълнени на който и да е процесор или графичен процесор. Повечето енджини за рендиране се разработват за един или повече API (Application Program Interface) като Direct3D и/или OpenGL, които предлагат слой за абстракция на софтуер за графичния процесор (GPU). Библиотеките от ниско ниво, като DirectX или OpenGL, са включени в игрите, защото предоставят независимост от хардуерен достъп до различни хардуерни средства. Тези хардуерни устройства могат да бъдат входни устройства като мишка, клавиатура и джойстик.

GR6. DLL files and Drivers/Device APIs - Файл с разширение DLL (библиотека за динамични връзки) е тип файл, който включва инструкции, написани под формата на програми, които могат да бъдат извикани или използвани от други програми за изпълнение на определени задачи. По този начин различните програми могат да споделят логически инструкции, имплементирани в един файл. Тези системни файлове играят поддържаща роля за изграждане на архитектурата на играта и помагат да се подобри нейната ефективност. API на устройството може да бъде дефиниран като, API (Application Programming Interface) който позволява на разработчиците да създават приложение което в крайна сметка може да взаимодейства с хардуерни устройства свързани или инсталирани на съответната система. Приложният програмен интерфейс (API) за устройства обикновено позволяват на крайните потребители да използват своя хардуер за взаимодействие със системата.

4.2.3. Идентифициране на външни системи

Външна система за играта стрелец от първо лице е само една и това е:

ES1. Web Service: Уеб услуга с помощта на която ще се записват резултатите на играча в реално време на външен сървър. За всеки играч който започва нова игра се генерират заявки който извикват услугата от външен сървър и записват постиженията му в отделен файл. Файловете с резултатите се генерират със специфичен формат като целта е данните да лесно четими и лесни за анализиране.

4.3. Нефункционални изисквания

Дефинирането на нефункционалните изисквания към играта е основна и важна задача преди да се анализират и извлекат изискванията към играта.

4.3.1. Изисквания за производителност и бързодействие

- Изискване за самата игра, като това е стандартно изискване е да изобразява сцените със скорост 60 fps (frames per second), което означава че 60 сцени трябва да могат да се изобразяват за 1 секунда или една сцена за 16.7 мили секунди.
- Друго изискване е в рамките на всеки frame трябва да може да се изпраща заявка към уеб услугата **с цел резултатите на играчът да се запишат на сървъра и подготвят за анализ.**

4.3.2. Изисквания за сигурност

- Трафикът между играта и уеб услугата трябва да е криптирана като целта да се защита на личните данни на играча.

4.3.3. Изисквания за надеждност и контрол на работата и дефектите

- Да записва всички възникнали проблеми по време на нейната употреба.
- При възникнал проблем поради логически проблем с кода на играта (бъг), необходимо е да се отстрани в рамките на 24 часа от нейното установяване и да се предостави новата версия.

4.3.4. Изисквания за преносимост

- Играта стрелец от първо лице (first-person shooter) трябва да може да се играе на операционна система Windows
- Да може да се играе на различни резолюции или казано така: да поддържа различни размери на екраните устройства
- Да поддържа следните уеб браузъри: Mozilla Firefox 42+, Microsoft Edge 42+

4.3.5. Изисквания за използваемост

- Играта трябва да е достъпна за всички потребители и в трите и варианта.

4.3.6. Изисквания за изпитаемост

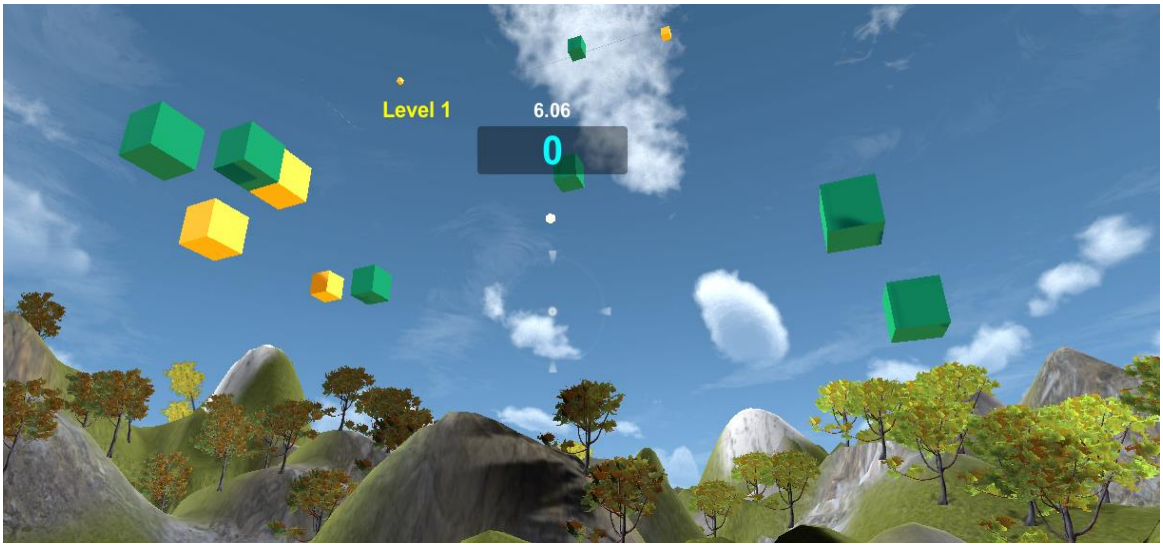
- Необходимо е играта да се тества преди да бъде завършена с цел проверка на това дали отговаря на първоначалните изисквания.

4.4. Преглед на видеоигра – за стрелци от първо лице

В тази точка ще направим цялостен преглед на играта като обърнем внимание на потокът на игра (game flow), геймплей (gameplay) на играта, механика, нива, интерфейс и др. Но първо ще обърнем внимание на концепцията на играта.

4.4.1. Кратко описание на играта (Box Shooter)

Box Shooter е single-player игра от типа FPS игри, като играчът е поставен във виртуална гориста среда. Играчът е заобиколен от движещи се обекти от различен цвят.



Фиг. 5 – Първо ниво, показващо видимото поле на главния герой

Целта на играта е да се съберат максимален брой точки за определен времеви интервал, като всеки цвят носи позитиви или негативи за играча. Зеления цвят носи точки за играча който се отчитат в средата на екрана, белия цвят увеличава времевия интервал а жълтия цвят отнема от времевия интервал.
За всяко ниво времевия интервал е различен, максималния брой точки е различен, както гъстота, скоростта, и посоката на движение на обектите е различна. Играта се играе в четири нива, като всяко ниво е различна сложност.

4.4.2. Нива в играта (Box Shooter)

На първо ниво броя на зелените обекти е по голям, движат се по бавно, броя точки необходим за завършване на нивото е по – малко, както и времето за достигането им. Видимостта е ясно и отчетлива.

Фиг. 6– Ниво 2, показващо видимото поле на главния герой

На второ ниво е нощ обектите са по-трудно различими, броя на зелените обекти е по-малък в сравнение с първо ниво, времето за достигане на крайната цел е по-малко, както и броя на точките за завършване на нивото е по-голям. На трето и четвърто ниво освен вариацията на тези параметри (време, точки, брой на различните обекти, скорост и посока на обектите) се добавя дъжд и мъгла.

Фиг. 7 – Ниво 3, показващо видимото поле на главния герой

Фиг. 8 – Ниво 4, показващо видимото поле на главния герой

Чрез вариациите на изброените параметри и добавяйки допълнителни компоненти към играта (дъжд, мъгла, нощ) се влияе върху сложността на играта.

4.4.3. Жанр на играта за стрелци от първо лице (Box Shooter)

Жанрът на играта е FPS (first-person shooter). Играчът използва движението на мишката за да контролира видимото поле на героя и съответно мерника, чрез кликвания на мишката играчът изстрелва куршум с който може да уцели избрания от него обект.

4.4.4. Контроли в играта

Това което е необходимо за да се играе играта е мишка с помощта на която се контролира видимото поле на героя за да се прицели и съответно да уцели чрез натискане на левия бутон избрания от него обект. Освен това се дава възможност на играча да използва и клавиатурата за контрол на героя.

Controls	Actions
Spacebar or Left Mouse Button	Shooting
Mouse Movement	Using mouse to move camera in all directions
Arrow Key - '↑'	Up
Arrow Key - '↓'	Down
Arrow Key - '←'	Right
Arrow Key - '→'	Left

Таблица. 5 – Описание на контроли за управление в играта

4.4.5. Интерфейс на играта

Средата на екрана се визуализират текущия брой точки достигнати по време на игровия процес, като над полето с точките се изписва времевия интервал за който играчът трябва да достигне необходимия брой точки, за да премине на следващо ниво. При достигане на необходимия брой точки, играта изписва съобщение „Level Complete“ и подканва играчът със съобщение „Play Next Level“ в долната част на екрана да премине на следващо ниво, чрез кликване на мишката върху съобщението.

4.4.6. Аудио ефекти

Играта е снабдена с музикално оформление подсказващо че трябва времето в играта тече бързо. При уцелването на обектите се издава звук който е различен за различните обекти, ако играча уцели обект носещ му точки звукът е подходящо подбран така че да подскаже че попадението е точно, ако уцели обект носещ му време звукът е друг, докато при уцелването на обект отнемащ му от времето, звукът е трети. Разбира се при стрелба се издава звук от играта наподобяващ лазер. В трето и четвърто ниво валящият дъжд също издава шум наподобяващ дъжд в реалния свят.

4.4.7. Прогресия в играта

Играчът трябва да завърши и четирите нива в играта за да завърши успешно играта, като за всяко ниво трябва да събере определен брой точки в рамките на поставеното му време, за да премине на следващо ниво. Всяко ниво предоставя на играча различна трудност.

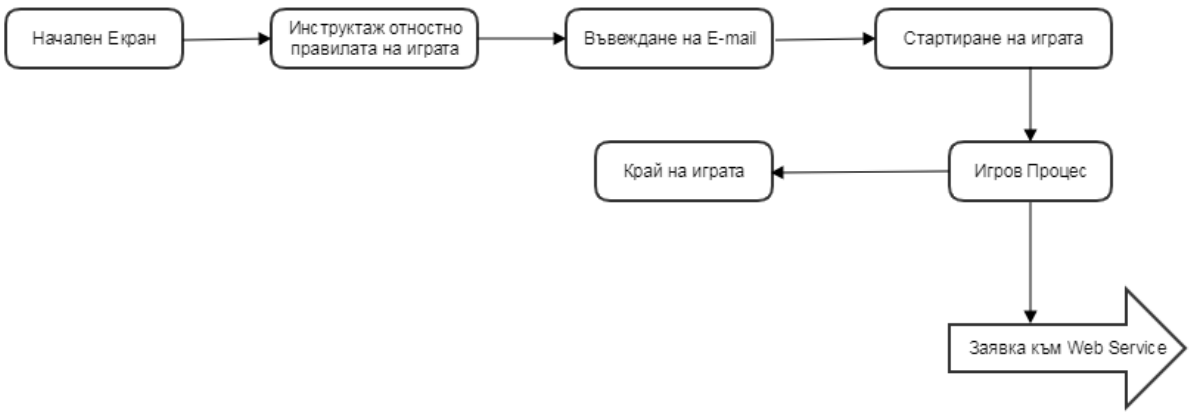
4.4.8. Физика в играта

Героят не може да извършва движения напред, назад и в страни, както не може да преминава или да се сблъсква с други обекти в играта, не може да лети или да скача, това което може е да се върти около оста си, да вдига погледа си или да свежда погледа си. Героя като цяло не се вижда, вижда се само мерника от неговото оръжие. Източника на светлина е симулативен и наподобява дневна или нощна светлина. Появата на мъгла и дъжд намалява видимостта спрямо хоризонта, и обектите около него. Обектите около играча могат да се препокриват, отдалечават, движат във всички посоки описващи прави линии на движение (горе - долу, ляво - дясно), могат да променят размера си, да се появяват и да изчезват. Камерата е от гледната точка на героя.

4.4.9. Механика на играта

Всяко едно ниво през което преминава играча, предоставя различно ниво на сложност. Героят е в центъра на играта, като от него се вижда само мерника на неговото оръжие в играта, движейки го във всички посока може да се прицелва в отделните обекти, които не могат да се сблъскват с героя но могат да се доближават до него, както и един до друг, освен това могат да се препокриват да се отдалечават или да се приближават, да се появят и да изчезват. Движението на мерника симулира движението на героя като това е движение около оста, нагоре и надолу. Взаимодействието на героя с обектите около него е посредством изстрел от оръжието си. Всеки друго обект като планини, дървета, облаци са част от декора.

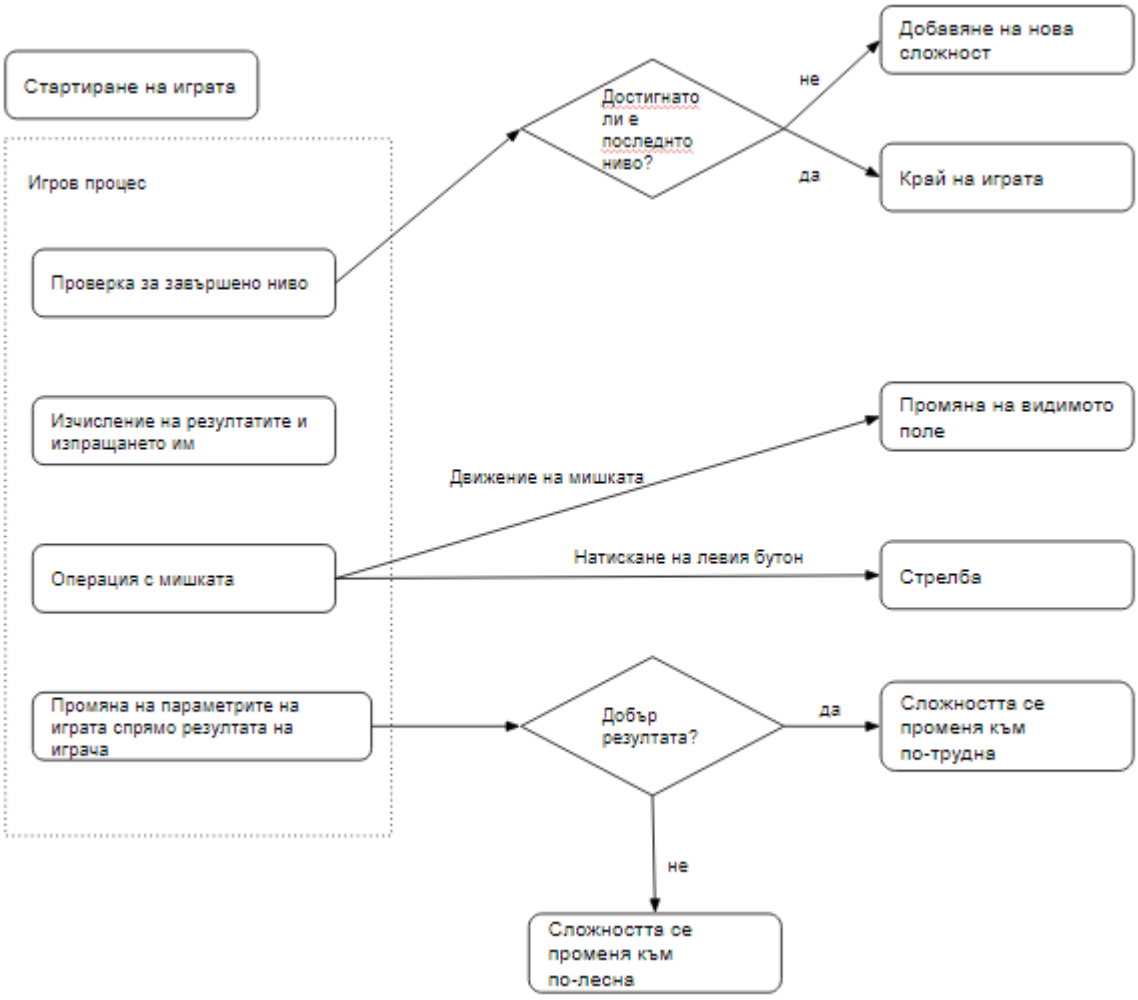
4.4.10. Поток на играта (Game Flow)



Фиг. 9 – Game Flow диаграма изобразяваща различните процеси през които минава играта – стрелба от първо лице

При стартиране на приложението играчът се подканва да въведе валиден e-mail адрес, като в същото време се предоставя описание с правилата на играта. След като е въведен имейл

адрес от негова страна играта започва. В зависимост от това коя версия е избрал играчът да играе, трудността на играта на различно ниво се променя по различен начин. Ако играчът игра неадаптивната версия, при всяко следващо ниво трудността се увеличава без да се променя по време на нивото. При първо ниво видимостта на играча от перспективата на героя е ясна, на второ ниво действието на играта се развива вечерно време, на трето ниво се появява дъжд, на четвърто ниво има в допълнение мъгла. При всяко следващо ниво броя на положителните обекти (тези според които играча печели точки или според тези с които може да увеличи времевия си интервал) се увеличават докато отрицателните обекти (тези които отнемат от времето му) намаляват за всяко ниво.
С други думи тези променливи са статични за съответното ниво. При адаптивната версия отново имаме тези променливи но те вече не са статични за всяко ниво на играта. Според резултатите на играча достигайки зададени вина в логиката на играта и спрямо кривите на учене игровия процес се изменя динамично за всяко ниво. В първо ниво времето е ясно и светло обектите са отчетливи и лесно различими с постоянна скорост, на второ ниво действието се развива през нощта като според резултатите на играча става по тъмно или по светло, като по този начин обектите са по – лесно или по – трудно различими, изменя и се скоростта на обектите както и броя им. На трето ниво се появява в допълнение на нощта и дъжд който изменя гъстотата си спрямо реликтите на играча, изменят се параметрите на обектите. На четвърто ниво освен нощно време, дъжд имаме и мъгла която също променя гъстотата си, освен тази на дъжда. Докато играчът играе играта, преминавайки през четирите нива, се изпращат уеб заявки към отдалече сървър на който е разположен уеб услуга където се записват резултатите на играча. След успешното приключване на четвърто ниво играта приключва и играчът излиза от приложението.



Фиг. 10 – Game Flow диаграма изобразяваща подробно игровият процес на играта – стрелба от първо лице

4.5. Изводи

Разглеждайки функционалните и нефункционални изисквания, механиката на играта, игровия процес, потокът, контролите в играта, физиката на играта, дизайна на играта, можем да установим първоначалната рамка на предвижданите функционалности. От направения анализ се вижда че функционалностите са свързани с това играчът да се справи с изискванията на играта (с цел успешното и завършване) в различни атмосферни условия, времеви условия и в същото време променяйки динамично трудността на играта спрямо показаните резултати в различните нива.

5. Проектиране на игра за стрелци от първо лице

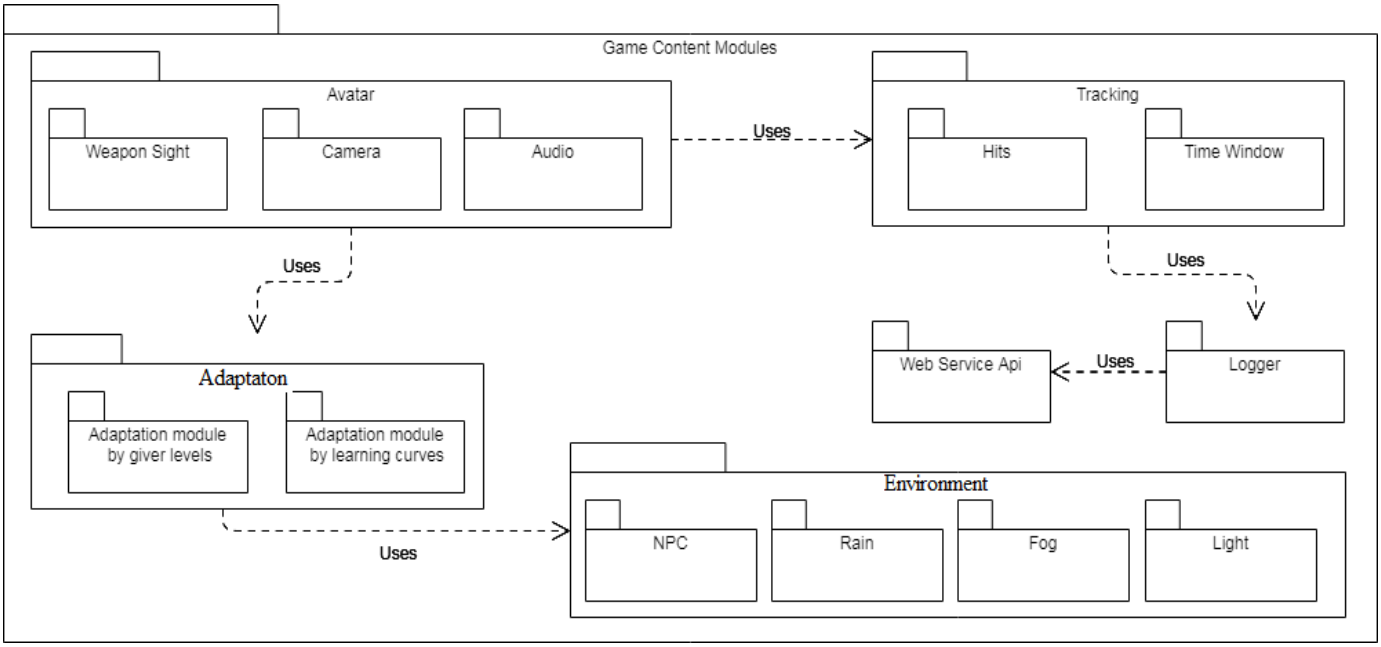
Основна част от реализацията на един краен продукт като игра е проектирането на самата игра обособявайки нейната архитектура. Идеята в създаването на тази архитектура е разделянето на бъдещата реализация на отделни модули според функционалните изисквания и в същото време да отговори на нефункционалните изисквания. В същото време, потребителския интерфейс е важна част от системата, за да може тя да бъде с адаптивен дизайн подходящ за устройства с различни екранни размери.

5.1. Архитектура

В тази точка ще разгледаме основните компоненти върху които се гради архитектурата на играта за стрелци от първо лице

5.1.1. Декомпозиция на модули и под модули, и употреба

Чрез модулна декомпозиция може да се направи логическо разделение на компонентите на играта на отделни единици за реализация. Това разделение помага за бъдещо предвиждане и по-лесно проследяване на промените, разделянето на работа по екипи, заменяемост на логически независимите модули от системата. Това са и основните мотиви за избора на документация на тази структура предвиждайки бъдеща поддръжка и последвала разширяемост.



Фиг. 11 – Модулна декомпозиция

На „Фиг. 11“ за изобразени отделните модули които са част от модулната декомпозиция на играта

Avatar – модулът който отговаря за превъплъщението на героя в играта и предоставя следните компоненти за неговото управление, мерника който е част от оръжието му, тъй като играта е тип FPS (First Person Shooter) играчът вижда само мерника на оръжието му, други детайли на аватара си не може да види, камерата което е от перспективата на героя и звука от изстрела който прави.

- **Weapon sight –** под модулът контролиращ движението на мерника и възпроизвеждането на изстрел, траекторията на куршума и съответно попаденията.
- **Camera –** под модулът отговорен за това героя да вижда виртуалния свят.
- **Audio –** този под модул е отговорен за издаването на звук при изстрел и съответно попадение, освен този звука от изстрела този модул се грижи за музиката в играта и шума от дъжда.

Tracking – Модулът който е отговорен за събирането на данни за попаденията по време на времевия прозорец за съответното ниво, този модул изпраща данните към модулът Logger

- **Hits –** под модул който отчита попаденията, отговорен е за това да разграничи всяко попадение като позитивно или негативно (да увеличи броя на точките, да увеличи времето за което е необходимо да се съберат съответния брой точки или да намали времето)
- **Time Window –** под модулът който отчита времевия прозорец в играта за съответното ниво.

Adaptation module – отговорен е за това да промени сложността на играта спрямо резултатите на играча и спрямо версията на адаптация. Промяната на сложността се осъществява като се променят характеристиките на околната среда, като гъстота на дъжда, като гъстота на мъглата, силата на светлината, скоростта на обектите обектите около героя в които се прицелва и тяхната гъстота.

- **Adaptation module by given levels** – тук промяната на сложността се прави на базата на нива на достигнат текущ резултат (брой попадения за време) спрямо предварително зададени такива, за всяко ниво на играта. При достигането на предварително зададените точки в играта за съответното ниво на игра, то сложността се променя.
- **Adaptation module by learning curve –** тук промяната на сложността се прави на базата на нива на достигнат текущи резултат (брой попадения за време) спрямо зададени криви на учене, за всяко ниво. Ако намери такива криви на учене, сложността на играта се променя.

Environment – модулът се занимава с физиката на всички обекти във виртуалния свят на играта. Грижи се за това да променя сложността на играта като променя заобикалящата го среда във виртуалния свят на базата на получените инструкции от модула за адаптация.

- **Light –** под модул чиято основна грижи е осветлението, промяната на деня в нощта, и за това колко ще е ясна нощта.

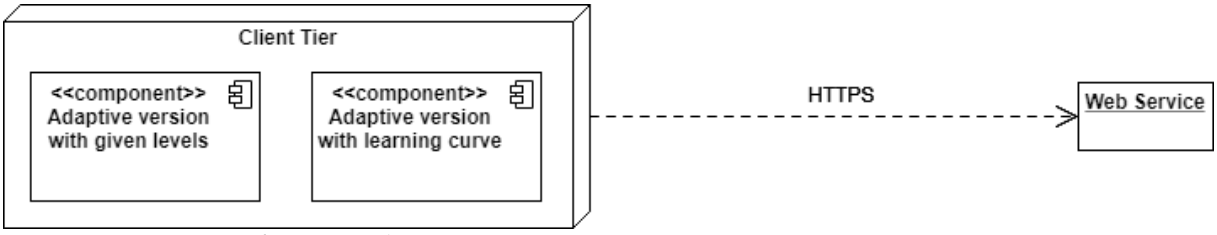
- **Fog –** под модул чиято основна грижи е мъглата, като може да променя нейната гъстота. Така видимостта на играча може да ще е по-добра или по-лоша.
- **Rain –** под модул чиято основна грижи е дъжда в играта, определя с каква сила ще вали и колко да е гъст дъжда.
- **NPC (non-person controller) –** под модул чиято основна грижи е динамиката на обектите, скоростта с която се движат, посоката на движение, тяхната гъстота, дали да се препокриват или не, цвета и формата на обектите както и големината им.

Logger – модул чиято основна грижи е обработката на получените данни от Tracking модула, изпраща обработените данни към външна за играта система и където се съхраняват за анализ.

Web Service API – външна системата, който получава изпратените данни от играта и ги записва на сървъра, в CSV файлов формат.

5.1.2. Структура на компонентите и конекторите

Структурата на компонентите и конекторите спомага за изобразяването на отделните процеси в системата и тяхната комуникацията помежду им.
С нейна помощ се предоставя възможност да се проследи разположението на отделните компоненти върху софтуер и хардуер които се явяват външни за играта.



Фиг. 12 – Изглед на компонентите и конекторите на системата

Според горната диаграма „Фиг. 12“ се вижда комбинирания изглед компонентите и конекторите, като елементите които структурата разглежда са:

Client tier – хардуер върху който е инсталирана една от версиите на играта.
Adaptive version with given levels – адаптивна версията на играта по зададени нива.
Adaptive version with learning curve – адаптивна версията на играта па криви на учене.
Web Service –уеб услуга на който се изпращат данните от различните версии на играта, и се записват на сървъра.

Описанието на връзките между компоненти:

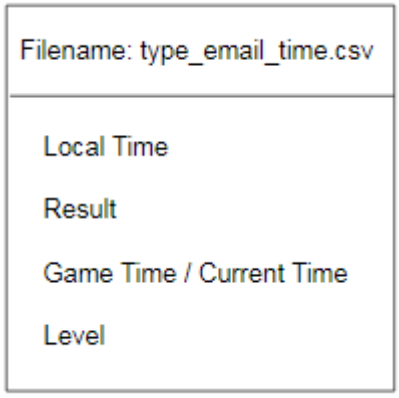
Client to a Web Service – Комуникацията се предава чрез HTTPS протокола, гарантиращ сигурността на данните.

5.2. Модел на данните

Общо прието определение за модел на данните е съвкупност от абстрактни понятия, чрез които се описват свойствата на разработеното приложение, на което са присъщи следните свойства:

- Статични свойства – задават се чрез обекти, техните атрибути и взаимовръзки помежду им
- Динамични свойства – определят се от операциите върху обектите

Определението за моделиране на данните се отнася до процеса на прехвърляне на нещата от реалния свят в света на компютрите. До началото на 90-те години на XX век компютрите се наричат Електронно Изчислителни Машини, като обработването на данни става посредством линейни структури от вида матрици, масиви и т.н. В днешно време релационните бази от данни безпроблемно осигуряват съхранение на този тип данни в хранилища, състоящи се от таблици, тест в бази данни, където съхраняват данни от компютърната система чрез специални структури и с помощта на таблици (един или няколко файла в зависимост от системата за управление на данни). Базите от данни е специфичен начин за оптимално съхранение на много голямо количество информация, която трябва да се използва многократно за решаване на много задачи от много потребители. Базата от данни прилича на файлова система на компютъра за съхранението на данни в компютъра чрез файлове, тест на физическо ниво тези два елемента са подобни. Съществената разлика се състой в начина на достъп и на обработване на съхранените данни. При файловата система достъпът до данните и тяхната обработка се осъществява чрез команди на операционната система. Пример за такъв достъп е приложение работещо под операционна система Windows е налице меню File, съдържащо команди за опериране с файлове.
При базите данни достъпа и тяхната обработка се осъществява чрез команда от специален език за програмиране, насочен към приложения в базите данни – SQL (език за структурни заявки).



Фиг. 13 – Структурата на файла съхраняващ резултатите на играча

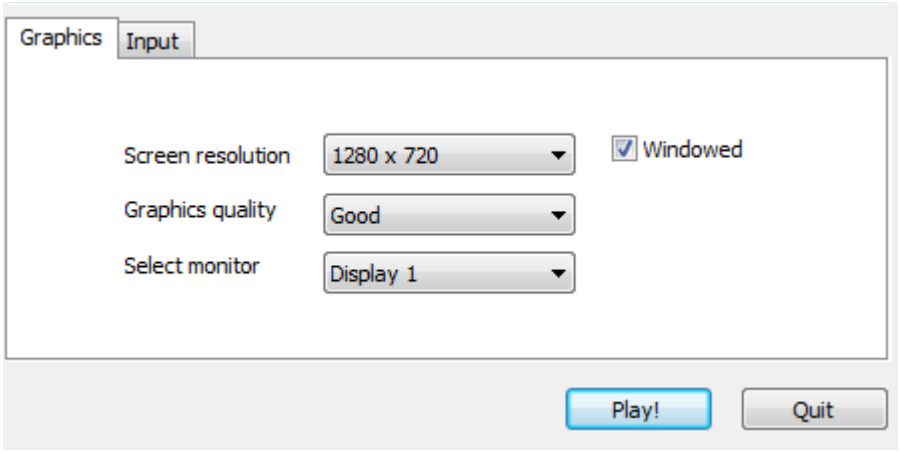
Играта стрелци от първо лице не използва релационни бази от данни за съхраните на данните (резултатите) от всеки играч, а използва файловата система на сървъра на който се помещава уеб услугата.
Както се вижда от „Фиг. 13“, структурата на файла се състой от четири колони в които се записват данни за:

- Локалното време – винаги е уникално
- Резултата на играча – броя точки
- Game Time – времеви прозорец за съответното ниво
- Level – текущото ниво на което играе играчът

За всеки frame играта изпраща HTTPS POST заявка със специфична структура от тип String:
String.Format("{0}; {1}; {2}; {3}; {4}", email, localTime, result, currentTime, level)
Уеб сървара от своя страна обработва заявката и запсва данните във файловата си система във файл с име типа на играта (адаптивна или не адаптивна), долна черта, емайла на играч, долна черта, времето в което е генериран файла в милисекунди и накрая разширение „CSV“.

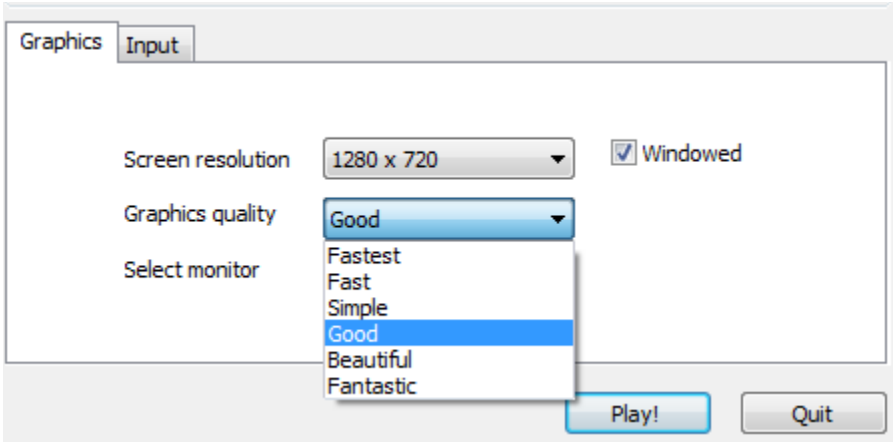
5.3. Потребителски интерфейс

Важен елемент в проектирането на играта е създаването на потребителски интерфейс който да е лесен за разбиране и интуитивен за самия потребител. Освен това играта трябва да може да се поддържа на различни по размер екрани. Играта трябва да може да работи на различни по вид уеб браузери. Задачата която трябва да бъде решена е проектирането и създаването на адаптивен дизайн. Създава се един и същ изходен код, който да взаимодейства с потребителя и да променя динамично подредбата и оразмеряването на обектите изобразени в играта според размера на екрана, като съществено значение има и бързодействието на играта. Това от своя страна води до едно доста добро изживяване на играча потапяйки се във виртуалния свят на играта, както и по-лесна поддръжка и последвала преизползваемост.



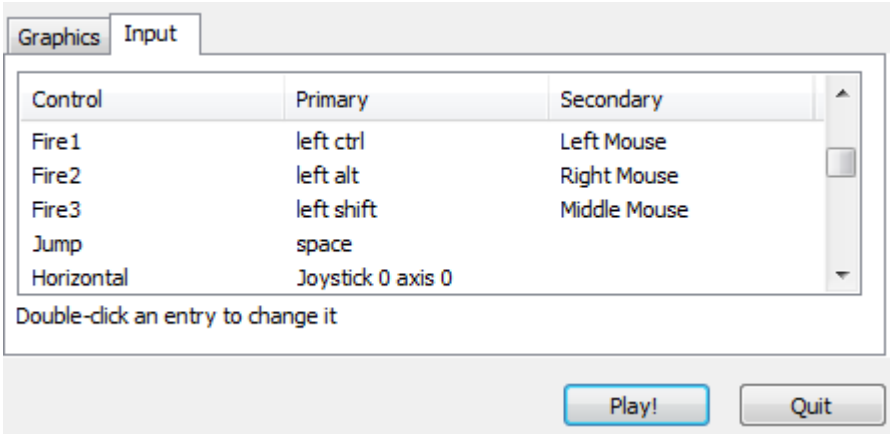
Фиг. 14 – Настройка на графиката

При стартирането на играта се дава възможност на играча да направи настройки на графиката. Позволява му се да промени резолюцията на екрана като по подразбиране винаги е избрана тази която е най-подходяща за компютъра на който ще играе.
Дава се възможност на потребителя да избере и качеството на графиката има няколко опции от който може да избира.



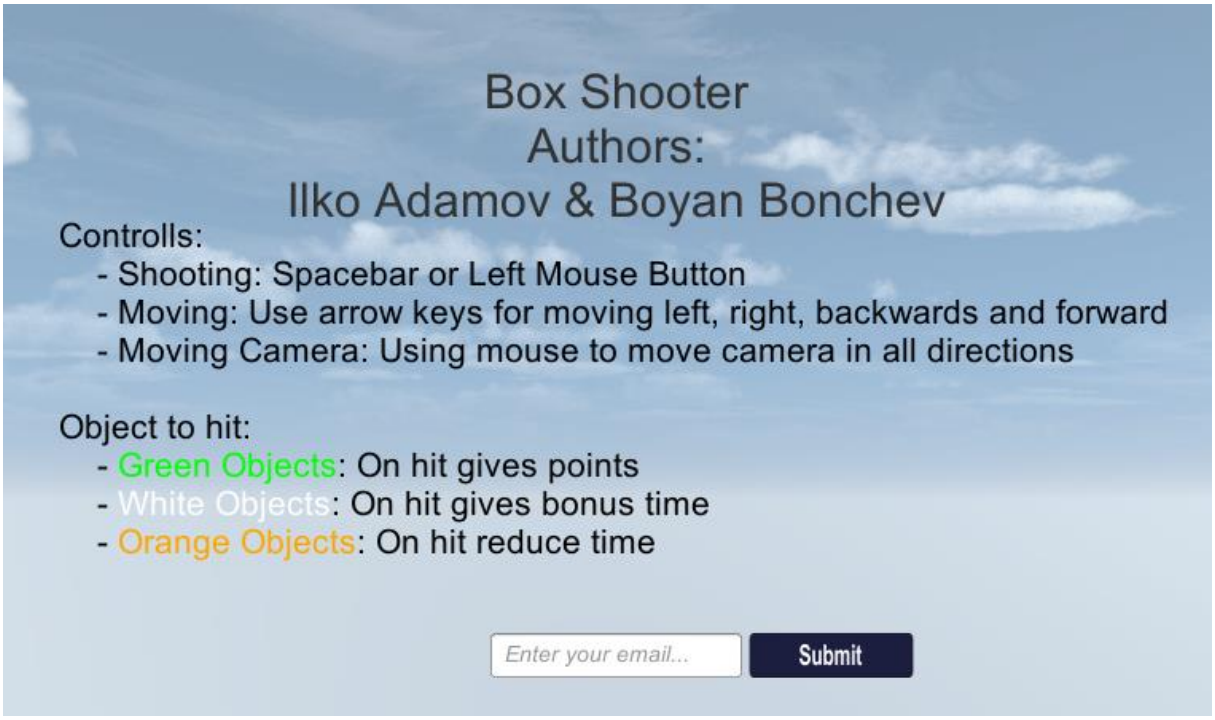
Фиг. 15 – Настройка на качеството на графиката

Опциите от който може да избира са: Fastest, Fast, Simple, Good, Beautiful, Fantastic. Зад всяка от тези опции стой две основни идеи, първата идея е графиката на играта да бъде колкото се може по реалистична и втората е свързана с бързодействието на играта. Ако компютъра е достатъчно мощен, избраната опция Fantastic би предоставила много по реалистична визия на играта и в същото време без да се налага играча да изчаква системата да направи всички изчисления по рендирането на обектите.
От „Фиг. 14“ можем да забележим че се дава възможност на играча да избира броя монитори на които да играе, за целите на дипломната работа играта е разработена да се играе само на един монитор. Освен всичко това потенциалният играч може да избира режимите на игра. Изборите са два, ако опцията Windowed не е селектирана то играта ще се стартира на цял екран (fullscreen), в противен случай играта ще се стартира в ограничен по размери според големината на екрана, прозорец.



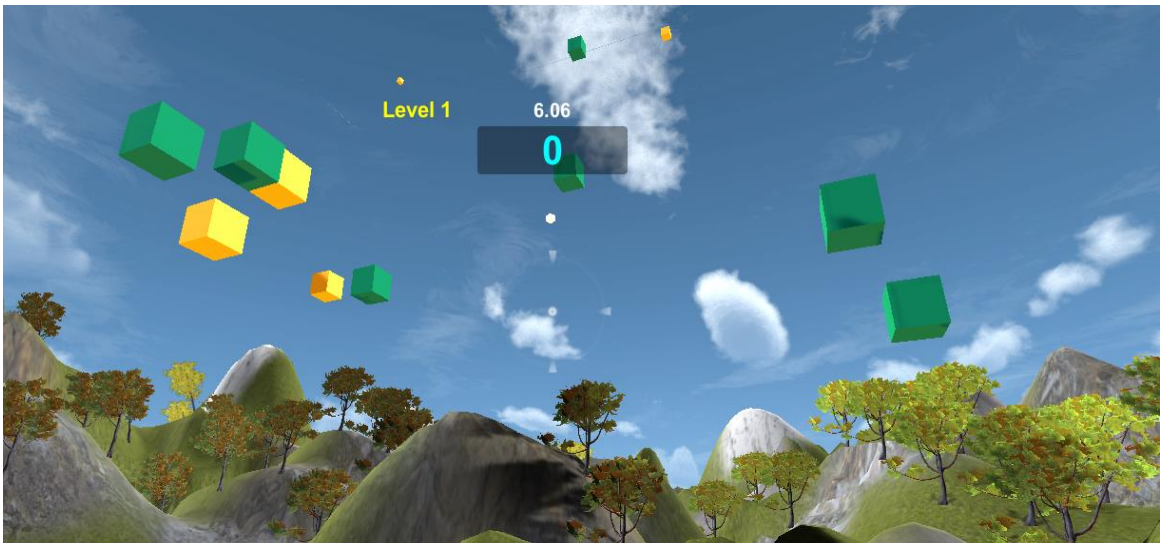
Фиг. 16 – Настройка на качеството на графиката

В таба "Input" от „Фиг. 16" са описани възможните контроли които играчът може да използва. Както се вижда предоставена е възможност да се играе и с джойстик.
След направените подходящи за играча настройки чрез бутан "Play!" може да започне играта.

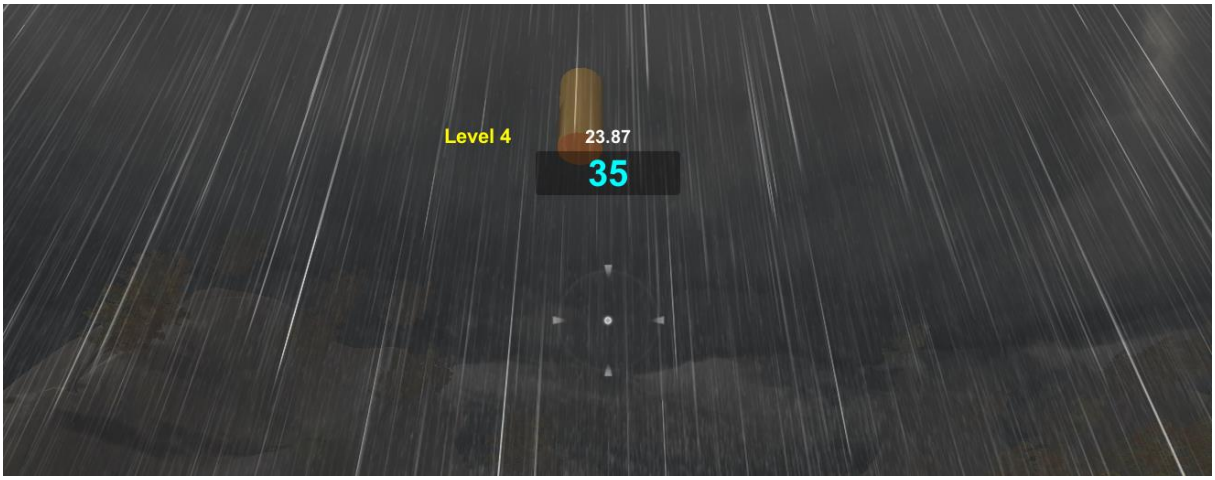


Фиг. 17 – Подготовка за стартиране на играта

В „Фиг. 17" са описани възможните контроли който може да използва играча в зависимост от неговите предпочитания. Дадена е информация за целите на играта, кой обект при уцелването му какви ползи и какви негативи носи. В Долната част на екрана е добавено текстово поле в което играчът е подканен на въведе имейл адрес след което да натисне бутона „Submit". При натискането на посочения бутон, играта започва от първо ниво.



Фиг. 18 – Игровия интерфейс първо ниво



Фиг. 19 – Игровия интерфейс четвърто ниво

От „Фиг. 18“ и „Фиг. 19“ се вижда че игровия интерфейс на играта предоставя информация за текущото състояния и постижения на играча. В средата на екрана се изобразява текущото постижение на играча, над него се изписва времевия прозорец за който играчът трябва да постигне определен брой точки за да премине на следващо ниво. До тези два индикатора се изписва нивото до което е достигнал играча. Под индикаторите за резултат и време е изрисуван мерника на героя.

Фиг. 20 – Игровия интерфейс за следващо ниво

От „Фиг. 20“ става ясно че при завършване на текущото нивото на мястото на индикатора за времеви прозорец се изписва „Level Complete“ което подсказва на играча че успешно е завършил съответното ниво, и около него обикаля подканващ го текст „Next Level“ за следващо ниво.

5.4 Изводи

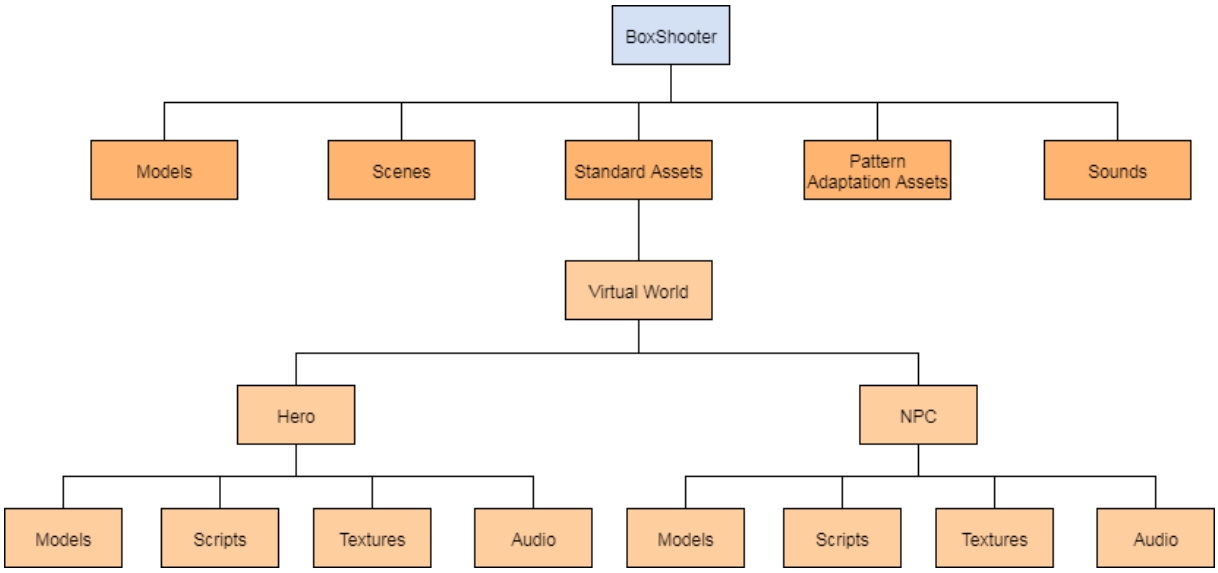
От всичко описано до тук се вижда че целта е да се специфицират изискванията, да се определят насоките, по които ще се разработва и реализира играта в следващите фази на развитие.
Разгледаната декомпозиция и нейната употреба е добро начало за бъдеща разширяемост на играта. Чрез ясното им разделение, скриване на вътрешната им структура, както и специфицирането на евентуални промени по модулната декомпозиция, се създават добри предпоставки за обособяването на отделните функционалности и аспекти на играта стрелци от първо лице, както и бъдещото и разширяване, без да се налагат промени или да се нарушава целостта и действието на вече съществуващите модули.

6. Реализация, тестване и внедряване

Позовавайки се на модулната декомпозиция, която беше описана в предишната точка вече знаем какво ще бъде логическото разпределение на кода и реализацията на играта, следвайки ясна зависимост между бизнес логиката и графичния интерфейс и ясното им логическо разграничаване. Всички модули ще комуникира и взаимодействат помежду си, в същото време ще бъдат достатъчно логически разделени. Платформата Unity която се използва за разработката на играта стрелци от първо лице предлага готови компоненти съставящи поставящи основите на играта. Unity предлага готови компоненти за терен (дървета, планини, небе – виртуалната околна среда на играта), мерник, дъжд, мъгла, светлина, различни аудио и звукови компоненти. По този начин се спестява моделирането на дизайна на играта и разработването на тези компоненти нужни за играта, като така акцента пада върху разработката на тяхното поведение във виртуалния свят, който ще се реализира с помощта на скриптова структура предлагана от Unity. Всеки скрипт може да се прикачи към съответния обект за който е предназначен, като по този начин всеки модул и под модул описан в модулната декомпозиция ще има съответния към него скрипт реализиращ неговата логика и това което трябва да прави.
На базата на така направения анализ се създава следния ред на разработка:

- Имплементация на управлението на героя
- Имплементация на аудио и звуковите ефекти
- Имплементация на контролните точки (точките на играча)
- Имплементация на времевия прозорец в играта
- Имплементация на различните методи на адаптация на сложност
- Имплементация на логиката за различните под модули на околната среда (дъжд, мъгла, светлина и др.)
- Имплементация на модула за изпращане на данни от играта към уеб услугата.
- Имплементация на уеб услугата
- Модулно тестване
- Компонентно тестване
- Системно тестване
- Unit test
- Acceptance test

6.1. Структура на проекта



Фиг. 21 – Дървовидна структура на играта

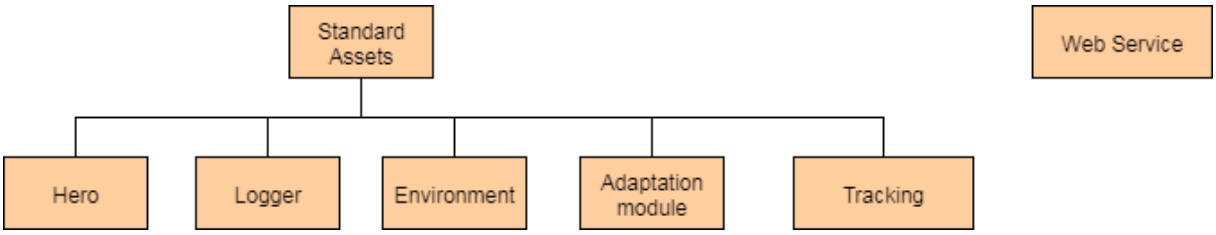
„Фиг. 21“ има за цел да опише структурата на проекта, физическото и логическото разделение на съставните части: модули, под модули, външни компоненти и др. Състой се от главна директория в която се съхраняват общите за употреба модули и под модули

- **Models** – Общите модели на елементи и обекти, намиращи се във виртуалния свят на играта. Това вида на обектите и елементите: как изглежда елемента, без приложените му характеристики, като цвят, материал, отражение, сянка.
- **Scenes –** тук се съхраняват различните сцени и нива на играта (начален екран, ниво 1 – 4, финален екран)
- **Pattern Adaptation Assets –** Директория на инсталираната библиотека и компонентите на кривата на учене (Player-centric rule-and-pattern-based adaptation asset). Използва се във версията на играта с адаптация по кривата на учене.
- **Textures –** Директория с различни текстури които се добавят към обектите и елементите
- **Sounds -** Директория на всички аудио файлове използвани от играта.
- **Standard Assets** – Директория в която се намират всички обекти във реалния свят (дървета, планини, дъжд, мъгла и др.)

Всяка вътрешна директория има сходна структура разделяйки различните компоненти на обекта, специфични само за него.

6.2. Реализиране на модулите

Реализацията на модулите разглежда техните интерфейси, връзка и интересни аспекти по време на имплементация. Това включва ключови методи, алгоритми механизми, които покриват функционалните и нефункционални изисквания към всеки отделен модул и под модул. На „Фиг. 22“ е показано физическото структуриране на модулите от декомпозицията.



Фиг. 22 – Модулите от системата

6.2.1 Реализация на модула на героя

Тук е реализирана логиката за управление на героя, тъй като изгледа е от перспективата на героя, тоест от самия герой не се вижда нищо освен мерника то погледа на героя съвпада с мерника, от където следва че движението на героя е според движението на мишката а от там и самата камера. С други думи движението на героя, мерника, камерата съвпада с движението на мишката.

```
void Start() {
              LockCursor (true);
              character = gameObject.transform;
              cameraTransform = Camera.main.transform;
              m_CharacterTargetRot = character.localRotation;
              m_CameraTargetRot = cameraTransform.localRotation;
       }
```

Фрагмент 1 – играта започва с фиксирана позиция на мерника

```
	void Update() {
		LookRotation ();

		if(Input.GetButtonDown("Cancel")){
			LockCursor (false);
		}

		if(Input.GetButtonDown("Fire1")){
			LockCursor (true);
		}
	}
```

Фрагмент 2 – промяна на поведението на мишката

Функция представена във „Фрагмент 2“ задава логиката на поведение на мишката, завъртане около оста на героя, натискане на бутона за стрелба

```
	private void LockCursor(bool isLocked)
	{
		if (isLocked)
		{
			Cursor.visible = false;

			Cursor.lockState = CursorLockMode.Locked;
		} else {
			Cursor.visible = true;
			Cursor.lockState = CursorLockMode.None;
		}
	}
```

Фрагмент 3 – мишката (мерника) съвпада със зоната на играта

```
        public void LookRotation()
        {
                float yRot = Input.GetAxis("Mouse X") * XSensitivity;
                float xRot = Input.GetAxis("Mouse Y") * YSensitivity;

                m_CharacterTargetRot *= Quaternion.Euler (0f, yRot, 0f);
                m_CameraTargetRot *= Quaternion.Euler (-xRot, 0f, 0f);

                if(clampVerticalRotation)
                        m_CameraTargetRot = ClampRotationAroundXAxis
(m_CameraTargetRot);

                if(smooth)
                {
                        character.localRotation = Quaternion.Slerp
(character.localRotation, m_CharacterTargetRot,
                                                                        smoothTime *
Time.deltaTime);
                        cameraTransform.localRotation = Quaternion.Slerp
(cameraTransform.localRotation, m_CameraTargetRot,
                                                                    smoothTime *
Time.deltaTime);
                }
                else
                {
                        character.localRotation = m_CharacterTargetRot;
                        cameraTransform.localRotation = m_CameraTargetRot;
                }
        }
```

Фрагмент 4 – Ротацията на героя

Във „Фрагмент 4“ е описана логиката на ротация на героя, както вече уточнихме, мерника, мишката, погледа на героя и камерата съвпадат понеже това е игра за стрелци от първо лице.

```
public class Shooter : MonoBehaviour {

	public GameObject projectile;
	public float power = 10.0f;

	public AudioClip shootSFX;

	void Update () {
		if (Input.GetButtonDown("Fire1") || Input.GetButtonDown("Jump"))
		{
			if (projectile)
			{
				GameObject newProjectile = Instantiate(projectile,
transform.position + transform.forward, transform.rotation) as GameObject;

				if (!newProjectile.GetComponent<Rigidbody>())
				{
					newProjectile.AddComponent<Rigidbody>();
				}

	newProjectile.GetComponent<Rigidbody>().AddForce(transform.forward *
power, ForceMode.VelocityChange);

				if (shootSFX)
				{
					if (newProjectile.GetComponent<AudioSource> ())
{
						newProjectile.GetComponent<AudioSource>
().PlayOneShot (shootSFX);
					} else {
						AudioSource.PlayClipAtPoint (shootSFX,
newProjectile.transform.position);
					}
				}
			}
		}
	}
}
```

Фрагмент 5 – логиката която е отговорна за стрелба

6.2.2. Реализация на „Logger“ модула

Това е мястото където събраните данни се изпращат на уеб услугата чрез HTTPS POST заявка и се записват в подходящ формат за по-късен анализ. Като формата на данните се определя по следния начин:

```
String.Format("{0}; {1}; {2};", score, formattedCurrentTime,
playAgainLevelToLoad)
```

Фрагмент 6 – формат на данните.

- score – моментния резултат на играча
- formattedCurrentTime – времевия прозорец
- playAgainLevelToLoad – индикатор за нова игра

```
private void LogMessageInCSV(string logMessage, string columnNames, string
fileName)
    {
        String fileNameToUpload = prefixFileName + "_" +
PlayerPrefs.GetString("email") + "_" + fileName + currentDate + ".csv";
        String logFileName = "http://huibg.000webhostapp.com/" + serviceFileName
+ ".php";
        StartCoroutine(logWebServiceCall(logFileName, fileNameToUpload,
columnNames, DateTime.Now.ToString("hh:mm:ss:ffff") + "," + logMessage));

    }

    IEnumerator logWebServiceCall(string logFileName, string fileName, string
columnNames, string logContent)
    {
        WWW w = new WWW(logFileName + "?filename=" + fileName + "&&columnNames="
+ columnNames + "&&logContent=" + logContent);
        yield return w;
    }
```

Фрагмент 7 – функции изпращане на форматираните данни към услугата

6.2.3. Реализация на модула за околната среда

Този модул се грижи за околната среда, като дъжд, мъгла, светлина както и тяхното изменение спрямо резултатите на играча. Тези всички компоненти за предоставени от Unity като те са вградени в играта, това ни освобождава от тяхното моделиране и създаване. Де факто това което трябва да направим е да се грижим за тяхното изменение спрямо резултата.

```
private void UpdateRain()
        {
            if (RainFallParticleSystem != null)
            {
                if (FollowCamera)
                {
                    var s = RainFallParticleSystem.shape;
                    s.shapeType = ParticleSystemShapeType.ConeVolume;
                                  s.angle = 0.0f;
                                  s.radius = 25f;
                                  s.length = 25f;
                                  RainFallParticleSystem.transform.position =
Camera.transform.position;
                    RainFallParticleSystem.transform.Translate(0.0f, RainHeight,
RainForwardOffset);
                    RainFallParticleSystem.transform.rotation =
Quaternion.Euler(0.0f, Camera.transform.rotation.eulerAngles.y, 0.0f);

                                  if (RainMistParticleSystem != null)
                    {
                        var s2 = RainMistParticleSystem.shape;
                         s2.shapeType = ParticleSystemShapeType.Box;
                        Vector3 pos = Camera.transform.position;
                        pos.y += RainMistHeight;
                        RainMistParticleSystem.transform.position = pos;
                    }
                }
                else
                {
                    var s = RainFallParticleSystem.shape;
                    s.shapeType = ParticleSystemShapeType.Box;
                    if (RainMistParticleSystem != null)
                    {
                        var s2 = RainMistParticleSystem.shape;
                        s2.shapeType = ParticleSystemShapeType.Box;
                        Vector3 pos = RainFallParticleSystem.transform.position;
                        pos.y += RainMistHeight;
                        pos.y -= RainHeight;
                        RainMistParticleSystem.transform.position = pos;
                    }
                }
            }
        }
```

Фрагмент 8 – Функция за промяна на дъжда

```
RenderSettings.sun.intensity += lightIntensityModifier;
```

Фрагмент 9 – Изменение на светлината.

```
RenderSettings.fogDensity += fogModifier;
```

Фрагмент 10 – Изменение на гъстотата на мъглата.

6.2.4. Реализация на модула „Tracking“

В този модул се проследяват точките и времевия прозорец за играта като те се променят в зависимост от това каква цел е улучил героя, съответно се изписват на екрана.

```
public void targetHit(int scoreAmount, float timeAmount)
    {
        score += scoreAmount;
        mainScoreDisplay.text = score.ToString();

        currentTime += timeAmount;

        if (currentTime < 0)
            currentTime = 0.0f;

        mainTimerDisplay.text = currentTime.ToString("0:00");
    }
```

Фрагмент 11 – Изменение на резултата и времевия прозорец

```
public class TargetBehavior : MonoBehaviour
{

	public int scoreAmount = 0;
	public float timeAmount = 0.0f;

	public GameObject explosionPrefab;
	void OnCollisionEnter (Collision newCollision)
	{
		if (GameManager.gm) {
			if (GameManager.gm.gameIsOver)
				return;
		}

		if (newCollision.gameObject.tag == "Projectile") {
			if (explosionPrefab) {
				Instantiate (explosionPrefab, transform.position,
transform.rotation);
			}

			if (GameManager.gm) {
				GameManager.gm.targetHit (scoreAmount, timeAmount);
			}

			Destroy (newCollision.gameObject);

			Destroy (gameObject);
		}
	}
}
```

Фрагмент 12 – Логика следяща поведението на обекта

На „Фрагмент 12“ е логически код който следи за поведението на обекта. Ако обекта е бил уцелен и играта все още не е приключила то се извиква кода описан във „Фрагмент 11“ и който се грижи да промени един от двата параметър, резултата или времевия прозорец в играта както и да ги изпише на екрана.

```
    private void TrackScoreResults()
     {
         difChanger.Score2 = score;
         difChanger.TimeForScore2 = inGameTimer;

         difChanger.CalculateScoreResult();

         difChanger.Score1 = difChanger.Score2;
         difChanger.TimeForScore1 = difChanger.TimeForScore2;
     }

     private void TrackTimeResult()
     {
         difChanger.CalculateTimeResult();

         difChanger.TimeToCompare = currentTime;
     }
```

Фрагмент 13– Проследяване на резултата на играча

Логиката във „Фрагмент 13“ се използва извиква за всеки фрейм на играта следейки времето и резултата на играта (броя точки който събира играчът за времевия интервал на съответното ниво)

6.2.5. Реализация на външния модул за уеб услугата

Този модул е външен за системата, той се отговаря за получаването на данните, тяхното записване на отдалечен сървър на който се намира самата уеб услуга. Уеб услугата записва данните във файл с разширение CSV формат

```
<?php
   $isFileExists = false;
   if(file_exists($_GET['filename']))
   {
      $isFileExists = true;
   }

   $logFile = fopen($_GET['filename'], "a") or die("Unable to open file!");
   if(!$isFileExists)
   {
      $txt = $_GET['columnNames'];
      fwrite($logFile, $txt);
      fwrite($logFile, "\n");
   }

   $txt = $_GET['logContent'];
   fwrite($logFile, $txt);
   fwrite($logFile, "\n");

   fclose($logFile);
```

Фрагмент 14 – Имплементация на уеб услугата.

6.2.5. Имплементация на модула за адаптация

Тук, в този модул е заложена логика за адаптиране на играта спрямо резултатите на играча. Адаптирането на трудността се осъществява на базата на следните логически фрагменти код заложени в играта. Като тези изчисления се използват освен за адаптиране на играта по нива така и за адаптиране по кривата на учене.

```
    public void CalculateScoreResult()
    {
        var result = (this.Score2 - this.Score1) / (DividerTwo *
(this.TimeForScore2 - this.TimeForScore1));
        if (result < ControlScoreForScoreResult)
        {
            this.ScoreResult = Low;
        }
        else
        {
            this.ScoreResult = High;
        }
    }
```

Фрагмент 15 – Имплементация на логика за изчисляване на резултата на играча

```
  public static string CalculateScoreResult(int score1, int score2, float
timeForScore1, float timeForScore2)
    {
        var result = (score2 - score1) / DividerTwo * (timeForScore2 -
timeForScore1);

        if (result < ControlScoreForScoreResult)
        {
            return Low;
        }
        else
        {
            return High;
        }
    }
```

Фрагмент 16 – Имплементация на логика за изчисляване на резултата на играча

Тъй като адаптацията зависи не само от резултата на играча а и времевия интервал на съответното ниво до което е достигнал играча, то също трябва да се вземе под внимание при определяне адаптацията на сложността.

```
    public void CalculateTimeResult()
    {
        if (this.TimeToCompare > this.CurrentTime)
        {
            this.TimeResult = Low;
        }
        else
        {
            this.TimeResult = High;
        }
    }
```

Фрагмент 17 – Задаване на нива спрямо времевия резултат

Кода описан във „Фрагмент 16“ и „Фрагмент 17“ се извиква от описаната логика във „Фрагмент 13“ – проследяване на резултата на играча която пък от своя страна се използва в Game Manager на играта за всеки фрейм на играта където става инициализацията на всички начално стойности
При вече направените изчисления може да променим нивото на трудност на за играча по зададени нива.

```
void Update()
    {
        if (!gameIsOver)
        {
                if (inGameTimer - trackTimer >= OnSecondsToTrack)
                {
                    trackTimer = inGameTimer;
                    TrackScoreResults();

                    TrackTimeResult();

                    ChangeDifficult(playAgainLevelToLoad);
                }
        }
    }

private void ChangeDifficult(string level)
    {
        SetModifiers();

        switch (level)
        {
            case "Level1":
                ChangeSpeed(difChanger.SpeedModifier);

                break;
            case "Level2":
                ChangeSpeed(difChanger.SpeedModifier);
                ChangeLight(difChanger.LightModifier);

                break;
            case "Level3":
                ChangeSpeed(difChanger.SpeedModifier);
                ChangeLight(difChanger.LightModifier);
                ChangeDensityRain(difChanger.RainModifier);
                break;
            case "Level4":
                ChangeSpeed(difChanger.SpeedModifier);
                ChangeLight(difChanger.LightModifier);
                 ChangeDensityRain(difChanger.RainModifier);
                ChangeDensityFog(difChanger.DensityFog);
                break;
            default:
                break;
        }
```

Фрагмент 18 – добавяне на трудност на играта по нива

От „Фрагмент 18“ се вижда че промяната на трудността на играта се осъществява като за всяко ниво на играта променяме по някой от следните параметри. На ниво 1 се променя скоростта на движение на обектите. На ниво 2 променяме скоростта на движение и интензитета на светлината. На ниво 3 се променя скоростта, светлината и гъстотата на дъжда. На ниво 4 се променя скоростта на движение на обектите, светлината, гъстотата на дъжда и гъстотата на мъглата появила се играта.

При метода за адаптация по кривата на учене първо се определят метриките на самите криви.

```
testMetricPattern = new PatternBasedAdaptationAsset();
                    FittingLine fittingLine1 = new FittingLine(0.8, 0.6);
                    testMetricPattern.RegisterPattern("Decrease AA 5% for 4s",
"Performance", PatternFeature.None, "t t+2000 t+4000", "x x*0.90 x*0.8",
fittingLine1);
                    testMetricPattern.RegisterPattern("Decrease AA 10% for 6s",
"Performance", PatternFeature.None, "t t+5000 t+10000", "x x*0.80 x*0.6",
fittingLine1);
                    testMetricPattern.RegisterPattern("Decrease AA 20% for 8s",
"Performance", PatternFeature.None, "t t+6000 t+12000", "x x*0.60 x*0.2",
fittingLine1);
                    testMetricPattern.RegisterPattern("Increase AA 5% for 10s",
"Performance", PatternFeature.None, "t t+5000 t+10000", "x x*1.10 x*1.2",
fittingLine1);
                    testMetricPattern.RegisterPattern("Increase AA 10% for 10s",
"Performance", PatternFeature.None, "t t+5000 t+10000", "x x*1.20 x*1.4",
fittingLine1);
                    testMetricPattern.RegisterPattern("Increase AA 20% for 10s",
"Performance", PatternFeature.None, "t t+5000 t+10000", "x x*1.40 x*1.8",
fittingLine1);
```

Фрагмент 19 – Инициализиране на кривите на учене.

След като бъдат изчислени резултата на играча по начин показан в предишните фрагменти, данните се сравняват с кривите на учене за да се определи по какви параметри ще се променя (адаптира) трудността на играта спрямо самия играч и неговите показатели.

```
patterns = metricPattern.SetMetric("Performance", (float)difChanger.DensityFog, t);
```

Фрагмент 20 – Логика за изменение на гъстота на мъглата.

В класа Player-centric rule-and-pattern-based adaptation asset е реализиран Event Handler който изпълнява логиката за промяна на параметрите на околната среда.

```
metricPattern.PatternEventHandler(patterns, gameObject);
```

Фрагмент 21 – Event Handler за промяна параметрите обект

6.3. Планирането на тестването

Същинското тестване на системата преминава през няколко етапа с цел гарантиране на функционалните и нефункционалните изисквания към играта който бяха описани и предварително зададени. Развитието на една игра преминава през много цикли от етапи. Софтуерът за игри е дефиниран, оценен, проектиран, построен, тестван, преминал през алфа версия, бета версия и накрая финалната версия. Цикълът на разработване от етапите често се описват последователно, но основните задачи често протичат паралелно особено по време на развитие на играта. Тестването може да се изпълни на всеки етап от цикъла на проекта. Целта на цялостното тестване на играта е да се идентифицират и коригират грешките възможно най-рано, така че продуктът на края да е качествен и функционален. Компаниите които разработват игри за масова употреба разполагат с ресурс (цели отдели), отговорен за тестването на техните продукти. Поради невъзможността да се преминат всички типове тестове и сценарии, са избрани тези които пряко засягат играта стрелци от първо лице и в същото време е в рамките на възможното, като целта е да се покрият в максимална степен качествените изисквания на играта. Планират се следният тип тестове: Модулно тестване, Компонентно тестване, Системно тестване, Тестване за надеждност (Unit test) и накрая Тест за приемане (Acceptance Test).

6.3.5. Модулно тестване

Модулното тестване засяга взаимодействието на всеки един модул със останалите. Пример за това е са тестове който се отнасят до адаптивните модули на играта спрямо резултатите му във съответното ниво. Модула отговарящ за изпращането на данните към външна услуга по време на игровия процес. Модула отговорен за възпроизвеждането на звук при колизия м/у целта и куршума изстрелян от героя. Всяко едно такова тестване протича по време на разработката на играта, както и при цялостното и завършване.

6.3.6. Компонентно тестване

Този тест налага разбиването на играта на малки тестови компоненти. Тестването на играта означава проверка на всяка част от нея, като тези части включват:

- Менюта и техните функции
- Анимация – усещането за движение, реализъм, скорост на кадрите
- Звукът и звуковите ефекти – звуковия ефект от колизията между куршума който героя е изстрелял и заобикалящите го обекти, всеки обект при удар има собствено звуково оформление също така, звука от падащия дъжд.
- Музиката – всяко ниво на играта има музикално оформление
- Камера – камерата в играта е от перспективата на героя която може да се движи в почти всички посоки с помощта на мишката, мащабирането на обектите около героя спрямо неговата визия, също така изгледа на околния свят от перспективата на героя.
- Виртуалния свят, отделните нива и отделните сцени.

- Атрибутите на играча (мерника на героя, таймера за времевия прозорец, брояча за текущия резултат)
- Движението на героя спрямо неговите атрибути.
- Условието играча да премине на следващо ниво (какви са те и дали са изпълнени?)
- Логиката на AI (Artificial Intelligence) тест управлението на участниците в играта (движещите се обекти - мишените) които не са под управлението на играча.
- Специалните ефекти в играта

Опциите на играта – Стартиране на играта, изход на играта, преминаване на следващо ниво, повторно изиграване на нивото при провал, режимите на игра от гледна точка на това дали е на цял екран или в отделен прозорец, резолюцията с която играчът иска да играе играта, качеството на графиката, контролите за управление. [23]

6.3.7. Тест за надеждност (Unit test)

Този тест ни гарантира, че най-малките елементи на играта работят според критериите на изискванията на играта. Това може да се отнася до определен компонент или конкретен механизъм на играта който може да бъде тестван изолиран от останалите. Това обаче не означава че този механизъм или компонент е проектиран изолирано. Чрез този тест е проверен механизмите за адаптация на играта (механизъм за адаптиране чрез нива и механизъм за адаптиране чрез отриване кривата на учене). Освен този механизъм, чрез Unit test е проверен и механизма за изпращане на данните към уеб услугата.[24]

6.3.8. Системно тестване

Системния тест се извършва когато играта се играе от самото начало до самия край. Първите системни тестове могат бъдат и симулирани, но в някакъв етап от разработването на играта ще е необходим да се създаде и прототип. Чрез прототипа ще трябва да преработим и да тестваме играта няколко пъти. Системния тест е шансът да отметнем колко се може повече критерии за качество, преди да поканим друг играч или играчи на за тест, за да се опитаме да симулираме възможно най-много различни ситуации и сценарии в които играта би могла да се провали. След като сме проверили всички критерии за качество на играта, сме готови за тестовете за приемане. [24]

6.3.9. Тест за приемане

Този тест отразява обикновена игрална сесия, в която играчите играят играта. Този тест е тясно свързан с точка: Анализ на резултатите от тестването и начина на отразяването им.

6.4. Анализ на резултатите

По време на процеса на разработка на играта стрелци от първо лице, са създадени три версии от играта. Първата версия е неадаптивната версия която има четири нива, всяко

ниво на играта е определена сложност. Втората версия на играта е също с четири нива но сложността се променя динамично при всяко следващо игрово ниво, тази динамична промяна се осъществява на базата на зададени параметри. При третата версия динамиката на играта се променя отново динамично но спрямо кривите на учене. Както отбелязахме вече промяната на сложността се осъществява като се промени някой от характеристиките на околната среда (гъстота на дъжда, гъстота на мъглата, интензитета на светлината, гъстотата на обетите който реално са и мишени за играча). Играта и в трите и версии е предоставена за тестване на случайни хора, като хората участвали в тестването попълват и въпросник. Целта на въпросника е участника в експеримента е да се определи от негова гледна точка дали такъв тип игри биха били полезни за него самия.
Тъй като данните който играта генерира са огромни по обем (за всеки фрейм играта изпраща заявка към уеб услуга за съхранение на резултатите), анализът ще бъде направен само върху четвърто ниво от трите версии на игрите. Причината за това е че това ниво включва характеристиките на играта за всяко предишно ниво. Производителността на всеки играч се изчислява по формулата: Performance = (Points(Tn) – Points(Tn-1)) / (Tn – Tn-1)
Points (Tn) е функция на броя точки от времето или с други думи това са точките които е придобил играча в момент Tn.
Points (Tn-1) е функция на броя точки от времето или с други думи това са точките които е придобил играча в момент Tn-1.
Формулата която се използва за изчисление на производителността представлява относителната разлика в нарастването на текущия резултат между два момента и за играта е управляващата метрика на базата на която ще се променя трудността на играта

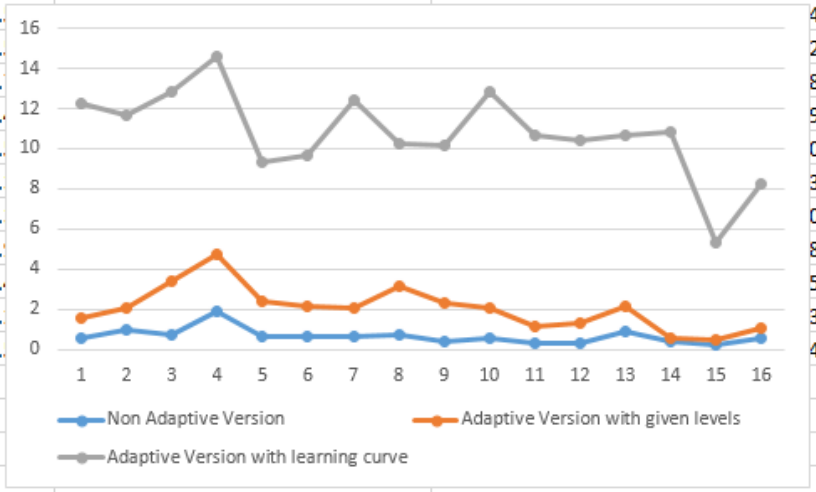
	A	B	C
1	Non Adaptive Version	Adaptive Version with given levels	Adaptive Version with learning curve
2	0.5098	1.0046	10.7282
3	0.984	1.1025	9.6153
4	0.7542	2.6153	9.4616
5	1.9257	2.7693	9.8709
6	0.6705	1.6778	7.0014
7	0.5971	1.5196	7.5409
8	0.5985	1.4296	10.4218
9	0.7471	2.3552	7.1835
10	0.4081	1.8574	7.8926
11	0.5652	1.4522	10.8016
12	0.3189	0.8417	9.5303
13	0.3373	0.9961	9.1039
14	0.9219	1.222	8.4838
15	0.4081	0.1782	10.2518
16	0.2193	0.2283	4.8365
17	0.5413	0.4913	7.2423

Фиг. 23 – Средни стойности за всеки играч за единица време от неадаптивната версия на играта, адаптивна версия на играта по зададени нива и адаптивна версия по крива на учене

В „Фиг. 23“ са събрани резултатите от всички участници до момента, за неадаптивната версия, адаптивната версия по зададени нива и адаптивната версия по кривата на учене. Резултатите от горе-посочената таблица представляват средна стойност от производителността за всеки играч, за единица време за въпросното четвърто ниво от трите игрите (неадаптивна версия, адаптивна версия със зададени нива и адаптивна версия по кривата на учене). Производителността на един играч в

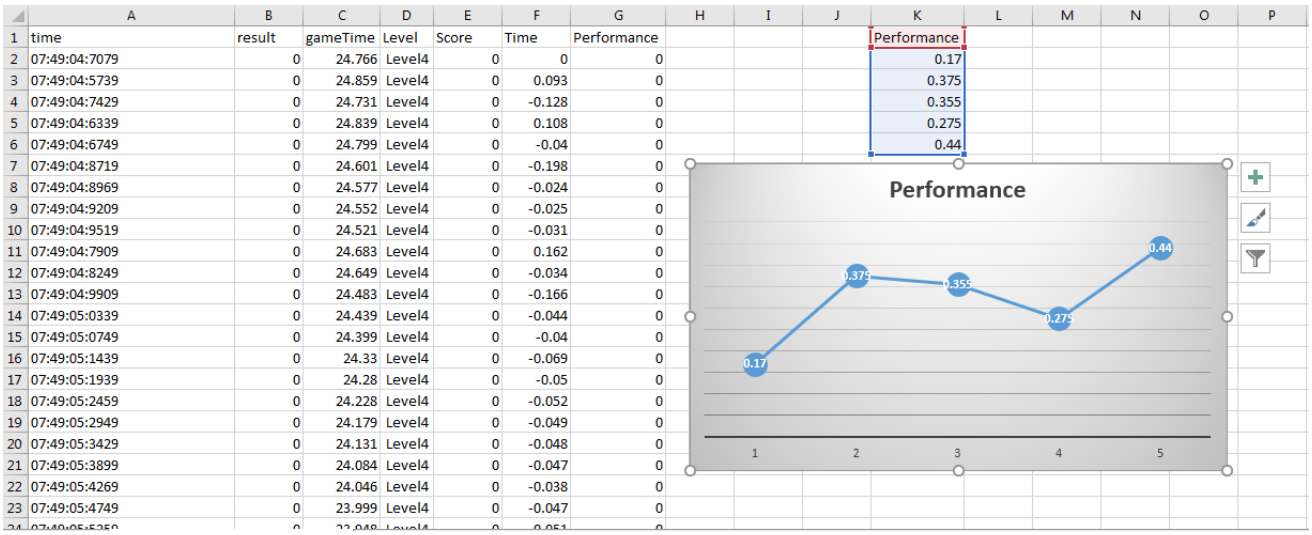
определен момент от игровия процес се изчислява чрез вече описаната логика във Фрагмент 16 (Имплементация на логика за изчисляване на резултата на играча)
За по-голяма изразителност данните се визуализират във вид на линейна графика

	A	B	C
1	Non Adaptive Version	Adaptive Version with given levels	Adaptive Version with learning curve
2	0.5098	1.0046	10.7282
3	0.984	1.1025	9.6153
4	0.7542	2.6153	9.4616
5	1.9257	2.7693	9.8709
6	0.6705	1.6778	7.0014
7	0.		409
8	0.		218
9	0.		835
10	0.		926
11	0.		016
12	0.		303
13	0.		039
14	0.		838
15	0.		518
16	0.		365
17	0.		423
18			
19			
20			
21			



Фиг. 24 – Средни стойности за всеки играч за единица време от неадаптивната версия на играта, адаптивна версия на играта по зададени нива и адаптивна версия по крива на учене, представени с линейна графика

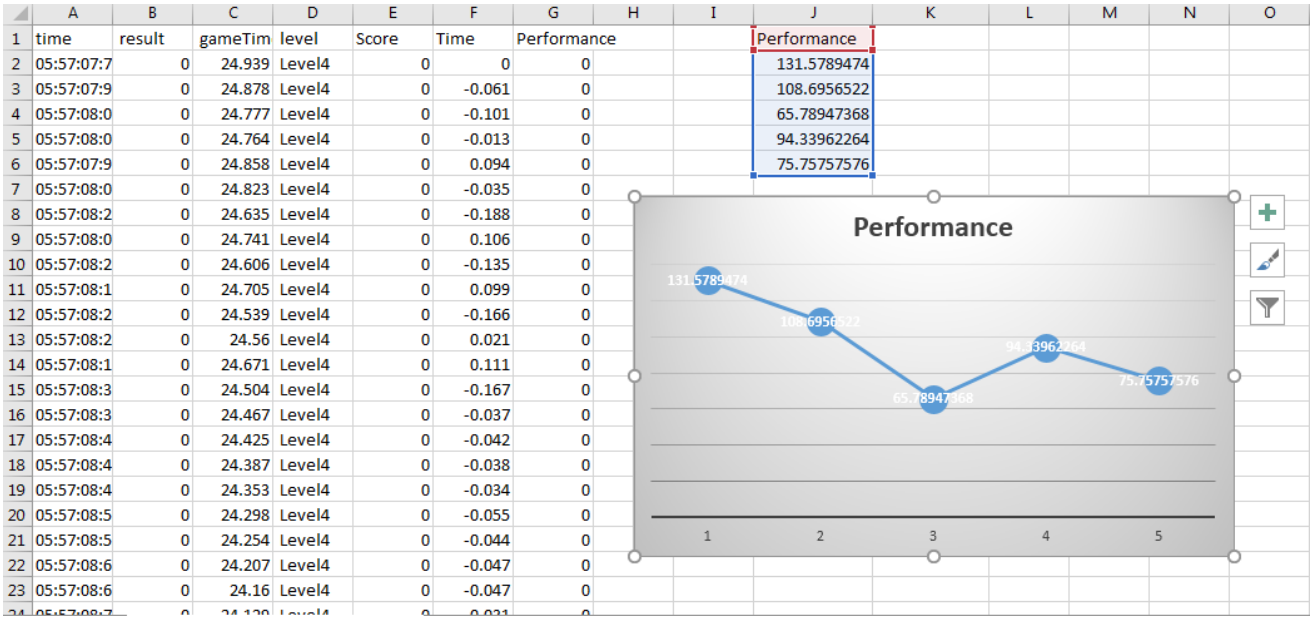
От графиката за средни стойности за всеки играч за единица време с линейна графика („Фиг. 24“) можем да направим следните заключения. От данните за производителност на неадаптивната версия се вижда че те са почти едни и същи без особени изменения, което ще рече че рано или късно играчът привиква към самата играта и след време тя ще му бъде безинтересна. От данните за адаптивната версия с зададени нива се вижда леко раздвижване в производителността, въпреки че играта е адаптирана по зададени нива то тези данни се приближават до данните на играта с не адаптирана версия. От данните за производителност на адаптираната версия на играта с кривата на учене се вижда разликата във динамиката спрямо другите две версии. Което от своя страна води до заключението че този подход на адаптация е доста по подходящ при адаптирането на игри спрямо играча.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	time	result	gameTime	Level	Score	Time	Performance				Performance					
2	07:49:04:7079	0	24.766	Level4	0	0	0				0.17					
3	07:49:04:5739	0	24.859	Level4	0	0.093	0				0.375					
4	07:49:04:7429	0	24.731	Level4	0	-0.128	0				0.355					
5	07:49:04:6339	0	24.839	Level4	0	0.108	0				0.275					
6	07:49:04:6749	0	24.799	Level4	0	-0.04	0				0.44					
7	07:49:04:8719	0	24.601	Level4	0	-0.198	0									
8	07:49:04:8969	0	24.577	Level4	0	-0.024	0									
9	07:49:04:9209	0	24.552	Level4	0	-0.025	0									
10	07:49:04:9519	0	24.521	Level4	0	-0.031	0									
11	07:49:04:7909	0	24.683	Level4	0	0.162	0									
12	07:49:04:8249	0	24.649	Level4	0	-0.034	0									
13	07:49:04:9909	0	24.483	Level4	0	-0.166	0									
14	07:49:05:0339	0	24.439	Level4	0	-0.044	0									
15	07:49:05:0749	0	24.399	Level4	0	-0.04	0									
16	07:49:05:1439	0	24.33	Level4	0	-0.069	0									
17	07:49:05:1939	0	24.28	Level4	0	-0.05	0									
18	07:49:05:2459	0	24.228	Level4	0	-0.052	0									
19	07:49:05:2949	0	24.179	Level4	0	-0.049	0									
20	07:49:05:3429	0	24.131	Level4	0	-0.048	0									
21	07:49:05:3899	0	24.084	Level4	0	-0.047	0									
22	07:49:05:4269	0	24.046	Level4	0	-0.038	0									
23	07:49:05:4749	0	23.999	Level4	0	-0.047	0									

Фиг. 25 – Извадка от резултатите за един играч на адаптивна версия на играта

На „Фиг. 25“ е предоставена извадка с резултати от адаптивната версия на играта за един играч. Данните които се записват са моментния резултат за единица време, времевия прозорец (оставащото време от текущия момент до края на нивото в рамките на което играчът трябва да достигне определен резултат).



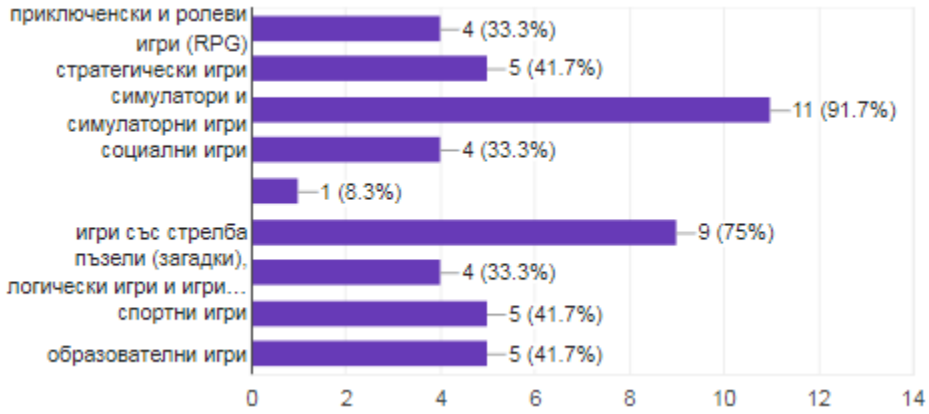
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	time	result	gameTim	level	Score	Time	Performance			Performance					
2	05:57:07:7	0	24.939	Level4	0	0	0			131.5789474					
3	05:57:07:9	0	24.878	Level4	0	-0.061	0			108.6956522					
4	05:57:08:0	0	24.777	Level4	0	-0.101	0			65.78947368					
5	05:57:08:0	0	24.764	Level4	0	-0.013	0			94.33962264					
6	05:57:07:9	0	24.858	Level4	0	0.094	0			75.75757576					
7	05:57:08:0	0	24.823	Level4	0	-0.035	0								
8	05:57:08:2	0	24.635	Level4	0	-0.188	0								
9	05:57:08:0	0	24.741	Level4	0	0.106	0								
10	05:57:08:2	0	24.606	Level4	0	-0.135	0								
11	05:57:08:1	0	24.705	Level4	0	0.099	0								
12	05:57:08:2	0	24.539	Level4	0	-0.166	0								
13	05:57:08:2	0	24.56	Level4	0	0.021	0								
14	05:57:08:1	0	24.671	Level4	0	0.111	0								
15	05:57:08:3	0	24.504	Level4	0	-0.167	0								
16	05:57:08:3	0	24.467	Level4	0	-0.037	0								
17	05:57:08:4	0	24.425	Level4	0	-0.042	0								
18	05:57:08:4	0	24.387	Level4	0	-0.038	0								
19	05:57:08:4	0	24.353	Level4	0	-0.034	0								
20	05:57:08:5	0	24.298	Level4	0	-0.055	0								
21	05:57:08:5	0	24.254	Level4	0	-0.044	0								
22	05:57:08:6	0	24.207	Level4	0	-0.047	0								
23	05:57:08:6	0	24.16	Level4	0	-0.047	0								

Фиг. 26 – Извадка от резултатите за един играч на неадаптивна версия на играта

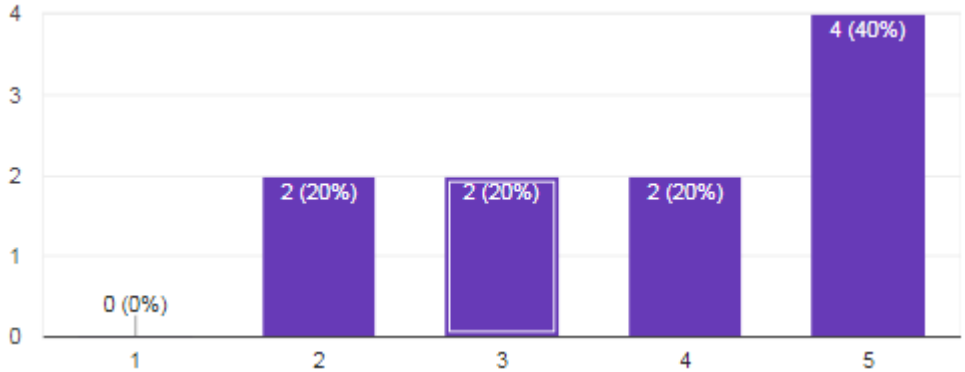
На „Фиг. 26“ е предоставена извадка с резултати от адаптивната версия на играта за един играч. Данните които се записват са моментния резултат за единица време, времевия прозорец (оставащото време от текущия момент до края на нивото в рамките на което играчът трябва да достигне определен резултат).

Полето Score се изчислява по формулата (Points(Tn) – Points(Tn-1)), като Points (Tn) е функция на броя точки от времето или с други думи това са точките които е придобил играча в момент Tn. Points (Tn-1) е функция на броя точки от времето или с други думи това са точките които е придобил играча в момент Tn-1.
Полето Time се изчислява по формулата (Tn – Tn-1), където Tn и Tn-1 представляват текущия и предишния момент в който играчът би трябвало да отчете точка (уцелвайки подходящия обект).

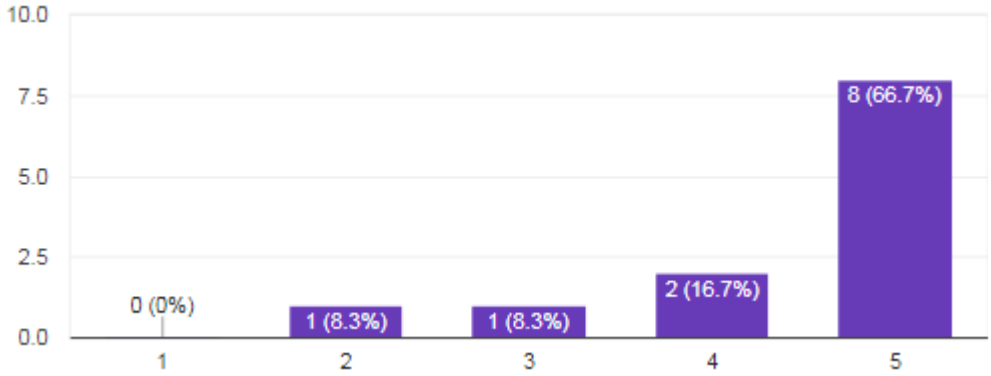
По долу са направени извадки от анкетата което съпровожда изследването, анкетираните са м/у възраст 7 и 47 години, като 75% от участвалите в анкетата са мъже, 25% са жени. Поради невъзможността да покажем всички графики ще приложим само тези който са по-важни за изследването (скалата по която са оценени отговорите са: 1 – Определено не, 2 – По-скоро не, 3 – Не мога да преценя, 4 – По-скоро да, 5 – Определено да)



Фиг. 27 – Предпочитани игри

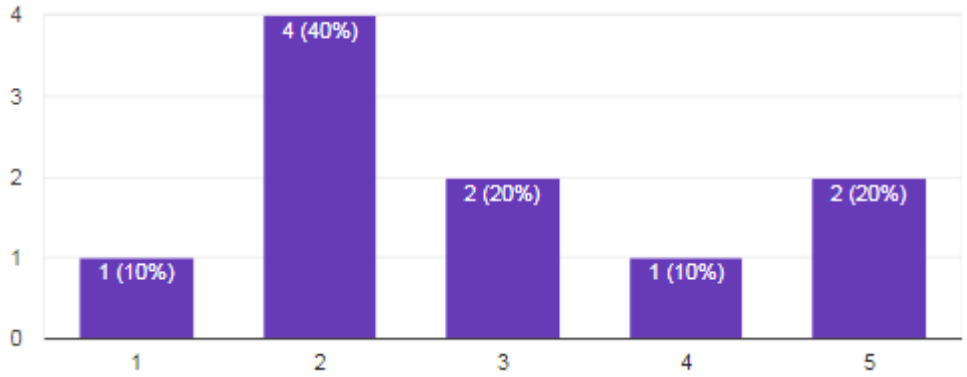


Фиг. 28 – Интерес към сценария (неадаптивна версия)

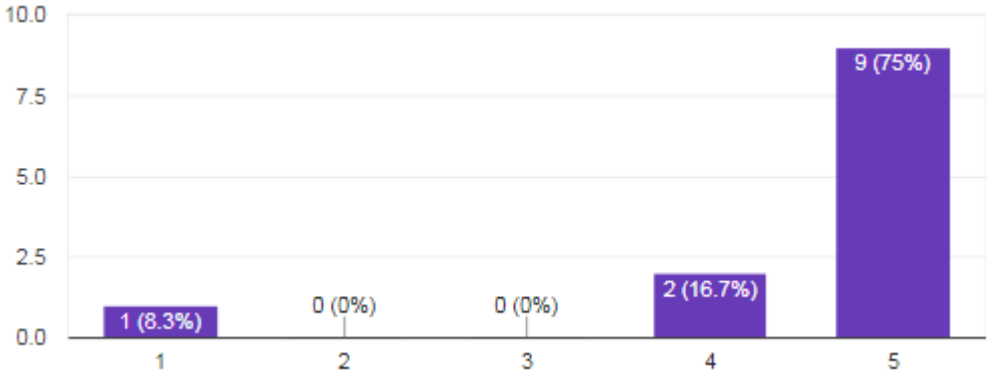


Фиг. 29 – Интерес към сценария (адаптивна версия)

От „Фиг. 28“ и „Фиг. 29“ се вижда че адаптивните игри са за предпочитане пред неадаптивните спрямо сценария на играта.

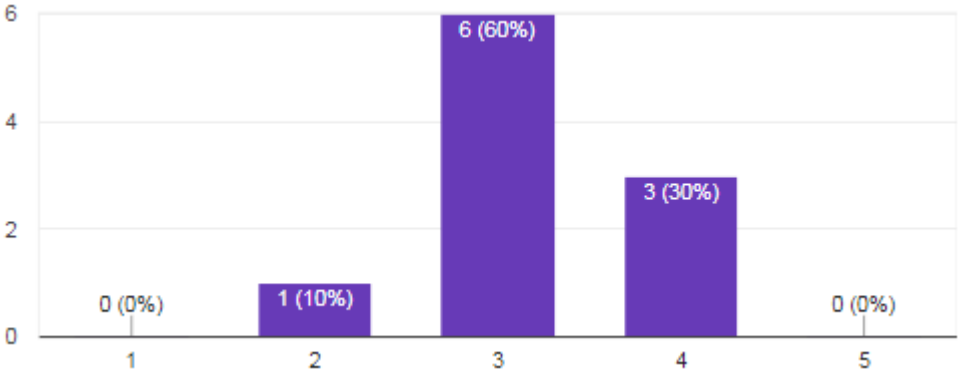


Фиг. 30 – Грешките в играта ми ме забавляваха (неадаптивна версия)

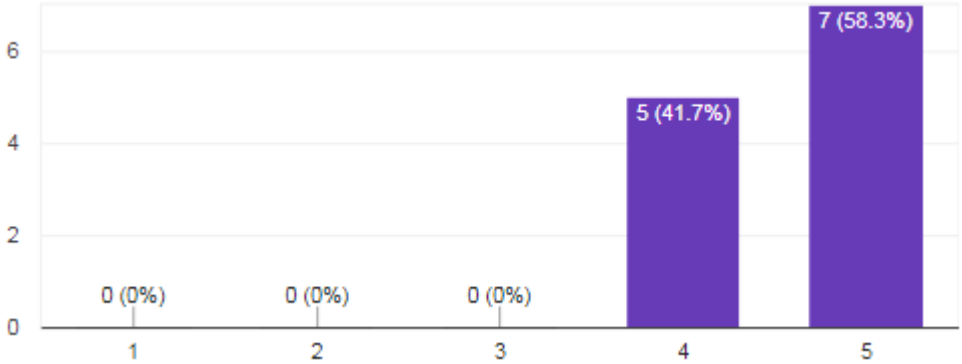


Фиг. 31 – Грешките в играта ми ме забавляваха (адаптивна версия)

От „Фиг. 30“ и „Фиг. 31“ се вижда че адаптивните игри са по забавни спрямо неадаптивните поради динамиката на играта.

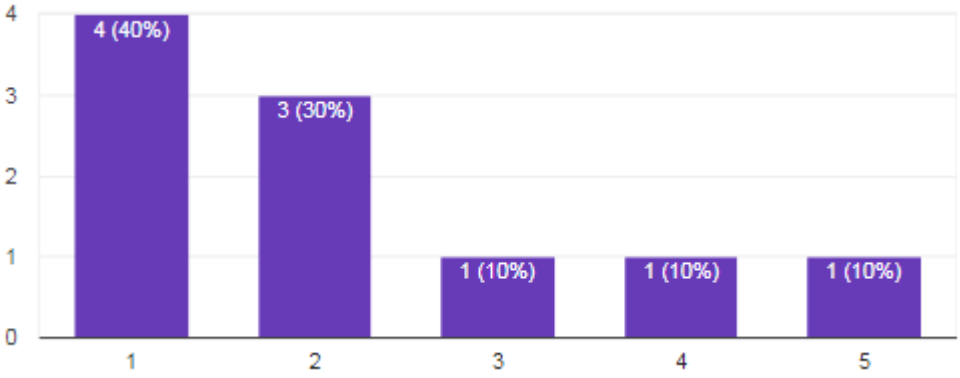


Фиг. 32 – Играенето бе приятно изживяване (неадаптивна версия)

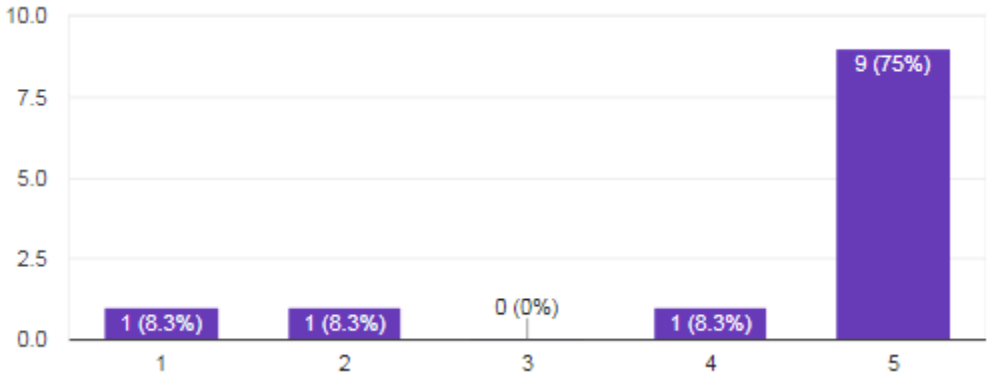


Фиг. 33 – Играенето бе приятно изживяване (адаптивна версия)

От „Фиг. 32“ и „Фиг. 33“ можем да заключим че адаптивните игри водят до приятни моменти.

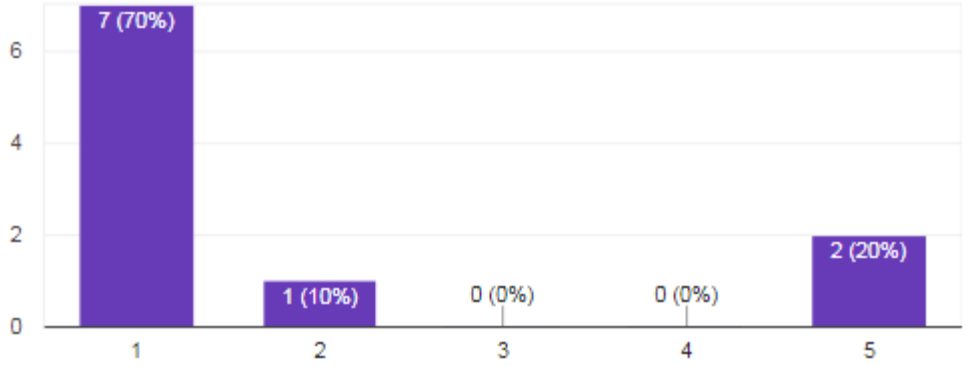


Фиг. 34 – Предизвикателството в играта (неадаптивна версия)

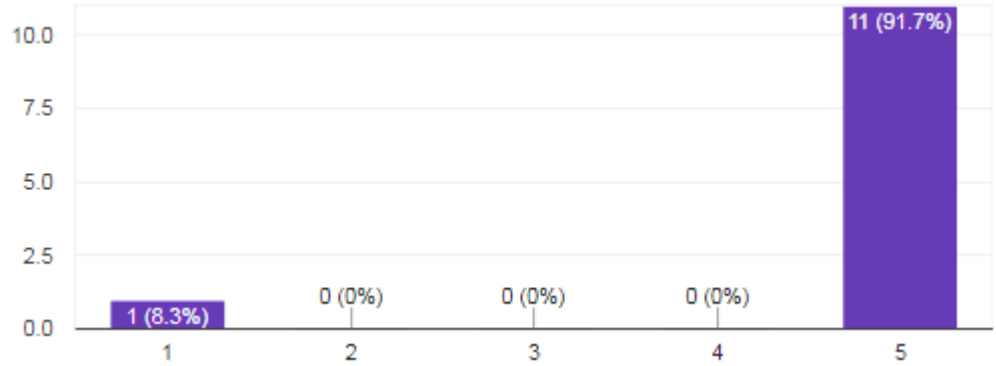


Фиг. 35 – Предизвикателството в играта (адаптивна версия)

От „Фиг. 34“ и „Фиг. 35“ можем да заключим че адаптивните игри са предизвикват играча



Фиг. 36 – Усетихте ли промяна на трудността в играта (неадаптивна версия)



Фиг. 37 – Усетихте ли промяна на трудността в играта (адаптивна версия)

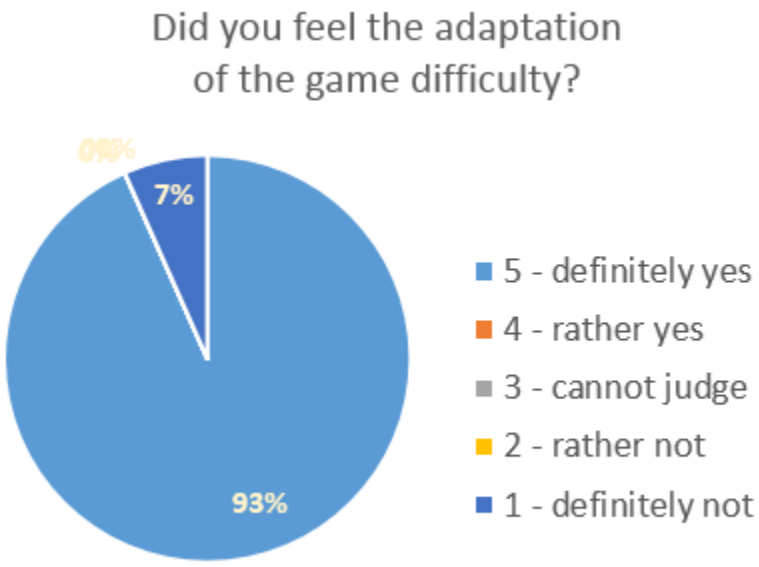
От „Фиг. 36“ и „Фиг. 37“ изводът е че играчът усеща динамиката на играта при адаптивните игри като че ли се пренася самия той във виртуалния свят на играта.

Използвайки теста на Стюдънт можем да кажем дали данните от анкетата са достоверни или не като приемем или отхвърлим нулевата хипотеза. Приемайки нулевата хипотеза то данни от анкета са валидни, ако я приемем то данни от анкетата са невалидни

t-Test: Two-Sample Assuming Unequal Variances		
	Variable 1	*Variable 2*
Mean	3.745736434	2.817531306
Variance	1.610424511	0.819955834
Observations	43	43
Hypothesized Mean Difference	0	
df	76	
t Stat	3.904281017	
P(T<=t) one-tail	0.000101511	
t Critical one-tail	1.665151353	
P(T<=t) two-tail	0.000203023	
t Critical two-tail	1.99167261	

Фиг. 38 – резултат от t-Test

На „Фиг. 38“ е резултата от изпълнението на теста на Стюдънт, на база всички средни стойности от отговорите от всеки въпрос от анкетите за адаптивната и неадаптивната версия на играта за всички участници. Ако стойността на променливата t Stat е по-голяма от отрицателната стойност на променливата t Critical two-fail, и t Stat е по-малка по стойност от положителната стойност на t Critical two-fail то приемаме нулевата хипотеза което ще рече, че данните от анкетите за адаптивната и неадаптивната версия на играта са невалидни. Ако обаче t Stat е по-малка по стойност от отрицателната стойността на t Critical two-fail или по-голяма от положителната стойност на t Critical two-fail, то можем да отхвърлим нулевата хипотеза, следователно данните от анкетите за адаптивната и неадаптивната версия на играта можем да приемем за валидни. В нашия случай t Stat е по-голяма по стойност от положителната стойност на t Critical two-fail, следователно отхвърляме нулевата хипотеза от което можем да направим извода че, данните от анкетата за адаптивната и неадаптиванта версия на играта са валидни.

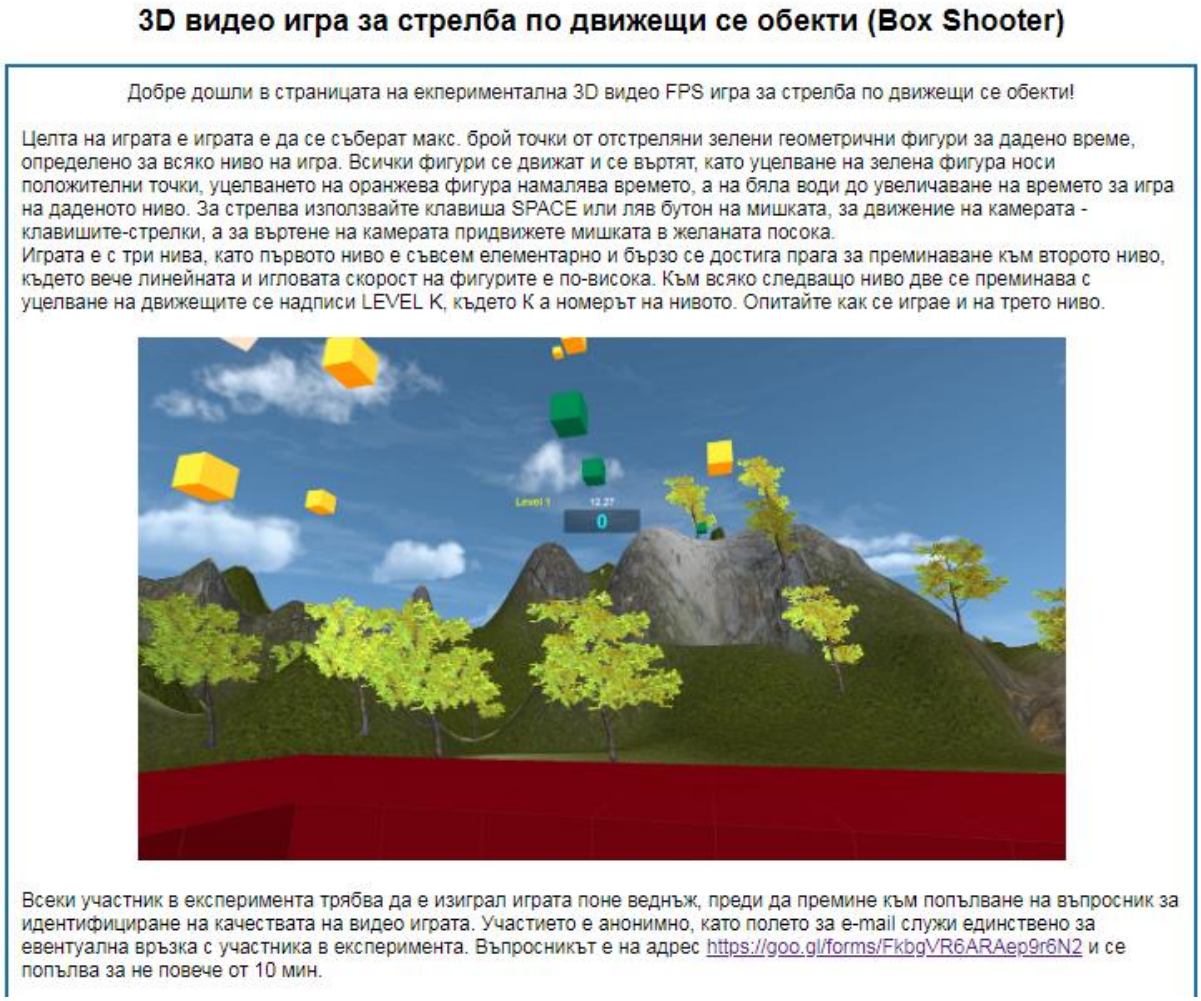


Фиг. 39 – Усетихте ли някаква промяна на трудността на играта (гъстота на мъглата, осветеност на терена и/или сила на дъжда)

„Фиг. 39“ е графично представяне на резултата от анкетата на въпроса: Усетихте ли някаква промяна на трудността на играта (гъстота на мъгла, осветеност на терена, сила на дъжда). От нея се вижда че 93% от анкетираните са усетили адаптацията на играта докато 7% не са. Тази извадка е направена според данните от анкетите за адаптивната версия и неадаптивната версия на играта който са доказани като валидни според теста на Стюдънт.

6.5. Експериментално внедряване

Внедряването е процес при който активно си сътрудничат клиента и вече реализирания продукт. Един софтуерен продукт се счита за внедрен когато е инсталиран на необходимите места. Адаптивната и неадаптивната версия на играта стрелци от първо лице са разположени на сървър любезно предоставен от проф. д-р Боян Бончев за целите на дипломната работа от където всеки желаещ любител на игрите може да участва в експеримента

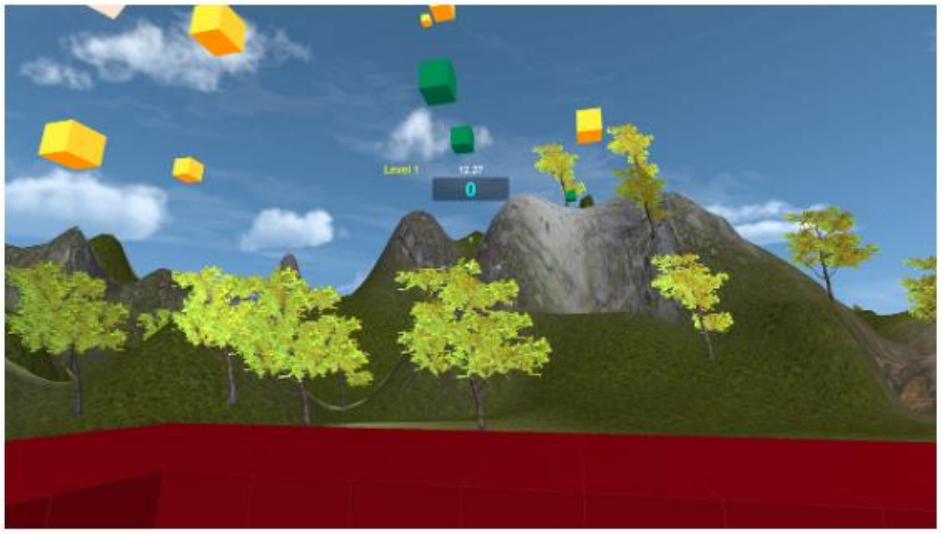


Фиг. 40 – Начална страница на неадаптивната версия на играта стрелци от първо лице

Адаптивна 3D видео игра за стрелба по движещи се обекти (Box Shooter)

Добре дошли в страницата на експериментална 3D видео FPS игра за стрелба по движещи се обекти, с адаптиране на динамичната трудност на задачата за стрелба на игровия процес!

Целта на играта е играта е да се съберат макс. брой точки от отстреляни зелени геометрични фигури за дадено време, определено за всяко ниво на игра. Всички фигури се движат и се въртят, като уцелване на зелена фигура носи положителни точки, уцелването на оранжева фигура намалява времето, а на бяла води до увеличаване на времето за игра на даденото ниво. За стрелва използвайте клавиша SPACE или ляв бутон на мишката, за движение на камерата - клавишите-стрелки, а за въртене на камерата придвижете мишката в желаната посока.
Играта е с три нива, като първото ниво е съвсем елементарно и бързо се достига прага за преминаване към второто ниво, където вече линейната и игловата скорост на фигурите е по-висока и се мени адаптивно, заедно с метеорологичните условия. Към всяко следващо ниво две се преминава с уцелване на движещите се надписи LEVEL K, където K а номерът на нивото. Опитайте как се играе и на трето ниво.



Всеки участник в експеримента трябва да е изиграл играта поне веднъж, преди да премине към попълване на въпросник за идентифициране на качествата на видео играта. Участието е анонимно, като полето за e-mail служи единствено за евентуална връзка с участника в експеримента. Въпросникът е на адрес https://goo.gl/forms/3aOcuDXgL1ArHlZH2 и се попълва за не повече от 10 мин.

Фиг. 41 – Начална страница на адаптивната версия на играта стрелци от първо лице

7. Заключение

7.1. Обобщение на изпълнение на началните цели

В обобщение на поставените цели пред настоящата дипломна работа трябва да обърнем внимание върху ключовите функционалности, които биха превърнали предмета на настоящата дипломна работа в работещ продукт. В днешно време има изключително много игри от най-разнообразни жанрове но голяма част от тях не предлагат адаптивност на сложността на играта нито по зададени нива нито по криви на учене, при тях нивото на сложност се задава още в началото. Поради тази причина настоящата дипломна работа, цели да покаже че човек сядайки пред компютъра пред играта може да усвои нови знания и умения.

По време на проучванията на технологичните решения при разработването на подобен тип игра, бяха разгледани и сравнени предимствата и недостатъците на най-популярните към момента платформи за разработка на игри. На базата на тези сравнения беше избрана платформата Unity за разработката на играта стрелец от първо лице.

На следващия етап от анализа на събраните изисквания бяха определени ключовите функционални и нефункционални изисквания. Също така беше подробно дефиниран потокът на играта при взаимодействието между играта и играчът.

Етапът проектиране на играта се вземат важни решения относно архитектурата на софтуера, както и нефункционалните изисквания към него. Тук се проектират връзките и комуникацията с различни модули и външни услуги, чрез декомпозиция на модули и под модули. Дефинират се логическото разделение на модули и под модули за разработка, като по този начин се обособяват основните и най-важни функционалности на играта.

Етапът за реализация е един от особено важните, тъй като без него няма реализиран продукт. Тук е моментът за осъществяване на направените технологични проучвания, анализираните изисквания и проектираната архитектура. На тази фаза се разискват и съответно имплементират алгоритми, заимстват се добри практики, интегрират се методите за адаптация в играта. Определят се стандартите за писане на код, които целят да направят кода на играта разбираем и лесен за разширение и поддръжка.

На етапа тестване която е неизменна част от разработката на играта се определят различните типове и нива на тестване, които да гарантират функционалните и нефункционалните изисквания към играта.

7.2. Насоки за бъдещо развитие и усъвършенстване

Играта стрелци от първо лице е игра която отчасти влиза в кръга „Сериозни игри“. Голяма част от хората не са чували за подобен тип игри а тези които са чували са слабо запознати. Причината за това е липсата на информация и разбира се финансови средства за разработка на подобен тип игри. Запознавайки се понятието „Сериозни игри“ почти всички участници заявяват, че според тях сериозните игри са подходящи за обучение и биха могли да се приложат в области на обучение където практическото обучение в реални условия е свързано с рискове, и че би им било интересно да играят такива игри и в бъдеще и че този тип игри биха дали по – добри резултати в сравнение с традиционните инструменти за обучение. Дипломната работа е разработена в рамките на научно-изследователския проект APOGEE (http://apogee.online), който планира да създаде решение, което от една страна да

запълни липсата на проста, но ефективна софтуерна платформа с отворен код за лесно изграждане на образователни видеоигри от не- ИТ специалист, а от друга страна – да отговори на нуждите от огромно количество специализирани адаптивни за обучение по различни учебни дисциплини. Получените данни от резултатите ще бъдат публикувани в статия за международна научна конференция.

Използвана литература

[1] **Wikipedia.** *Сериозна игра*
Свалено от:
https://bg.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B8%D0%BE%D0%B7%D0%BD%D0%B0_%D0%B8%D0%B3%D1%80%D0%B0

[2] **Wikipedia.** *Компютърна игра*
Свалено от:
https://bg.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%8A%D1%80%D0%BD%D0%B0_%D0%B8%D0%B3%D1%80%D0%B0

[3] **Bontchev, B., Vassileva, D.** *(2018) Dynamic game adaptation based on detection of behavioral patterns in the player learning curve, Proc. of the 10th Annual Int. Conf. on Education and New Learning Technologies, Palma de Mallorca (Spain), July 2018, ISBN: 978-84-09-02709-5.*

[4] **Bontchev, B., Vassileva, D.** *(2018) Detection of player learning curve in a car driving game, Proc. of 12th Annual Int. Technology, Education and Development Conference (INTED'2018), March 2018, IATED, Valencia, Spain, ISBN: 978-84-697-9480-7, pp.9302-9311.*

[5] **Bontchev, B., Vassileva, D., Ivanov, D.** *(2018) Player-centric adaptation of a car driving video game, Proc. of e-Society'18 Int. Conf., April 2018, IADIS, Lisbon, Portugal, ISBN: 978-989-8533-75-3, pp.193-200*

[6] **Bontchev, B. P., & Vassileva, D.** *(2017). Affect-based adaptation of an applied video game for educational purposes. Interactive Technology and Smart Education, Emerald, ISSN: 1741-5659, 14 (1), pp.31-49. DOI: 10.1108/ITSE-07-2016-0023*

[7] **Bontchev, B.** *Adaptation in Affective Video Games: a Literature Review, Journal of Cybernetics and Information Technologies, Vol. 16, No.3, 2016, pp.3-34, DOI: 10.1515/cait-2016-0032*

[8] **Bontchev, B.** *(2016-2017) Design of computer video games – Game Development*

[9] **Vassileva, D.** *Realizing and Applied Gaming Ecosystem*

[10] **NIT – New Internet Technologies.** *https://www.online-learning.bg/seriozni-igri.*
Свалено от: . *https://www.online-learning.bg/*

[11] **Steam Powered.** *Red Alliance*
Свалено от: *https://store.steampowered.com/app/594050/Red_Alliance/*

[12] **Steam Powered.** *Fallout 4*
Свалено от: *https://store.steampowered.com/app/377160/Fallout_4/*

[13] **Steam Powered.** *Left 4 Dead 2*
Свалено от: *https://store.steampowered.com/app/550/Left_4_Dead_2/*

[14] **Steam Powered.** *Doom*
Свалено от: *https://store.steampowered.com/app/379720/DOOM/*

[15] **Steam Powered.** *Sniper Ghost Warrior 3*
Свалено от: *https://store.steampowered.com/app/368070/Sniper_Ghost_Warrior_3/*

[16] **Constructor.** *https://www.scirra.com/*
Свалено от *https://www.scirra.com/*

[17] **Game Maker Studio.** *https://www.yoyogames.com/gamemaker*
Свалено от *https://www.yoyogames.com*

[18] **Unity.** *https://unity3d.com/*
Свалено от *https://unity3d.com/*

[19] **Unreal Engine.** *https://www.unrealengine.com/en-US/what-is-unreal-engine-4*
Свалено от *https://www.unrealengine.com/*

[20] **Microsoft.** *https://docs.microsoft.com/en-us/windows/desktop/ipc/named-pipes*
Свалено от *https://docs.microsoft.com/*

[21] **Sockets.** *https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-create-a-socket*
Свалено от *https://docs.microsoft.com*

[22] **Web Services.** *http://doveltech.com/innovation/the-pros-and-cons-of-web-services/*
Свалено от *http://doveltech.com*

[23] **Linked In**. *Game Testing Strategy*
Свалено от*: https://www.linkedin.com/pulse/game-testing-strategy-ilieana-petronova*

[24] **Board Game Geek.** *Testing Methodology Applied to Games*
Свалено от: *https://boardgamegeek.com/thread/1228786/testing-methodology-applied-games*

Bontchev, B. *(2016-2017) Design of computer video games – User Interface and experience*

Bantam Menace. *Game Design Document Version 1.1*
Свалено от: *http://www.photonstorm.com/downloads/CSC_GDD.pdf*

Digital BG. *https://www.digital.bg/kakvo-vliqnie-imat-video-igrite-varhu-nas-article435595.html.* Свалено от: . *https://www.digital.bg/*