**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: Екшън игра от първо лице във Виртуална Реалност

Дипломант: Научен ръководител:

*Даниел Янев Виктор Кетипов*

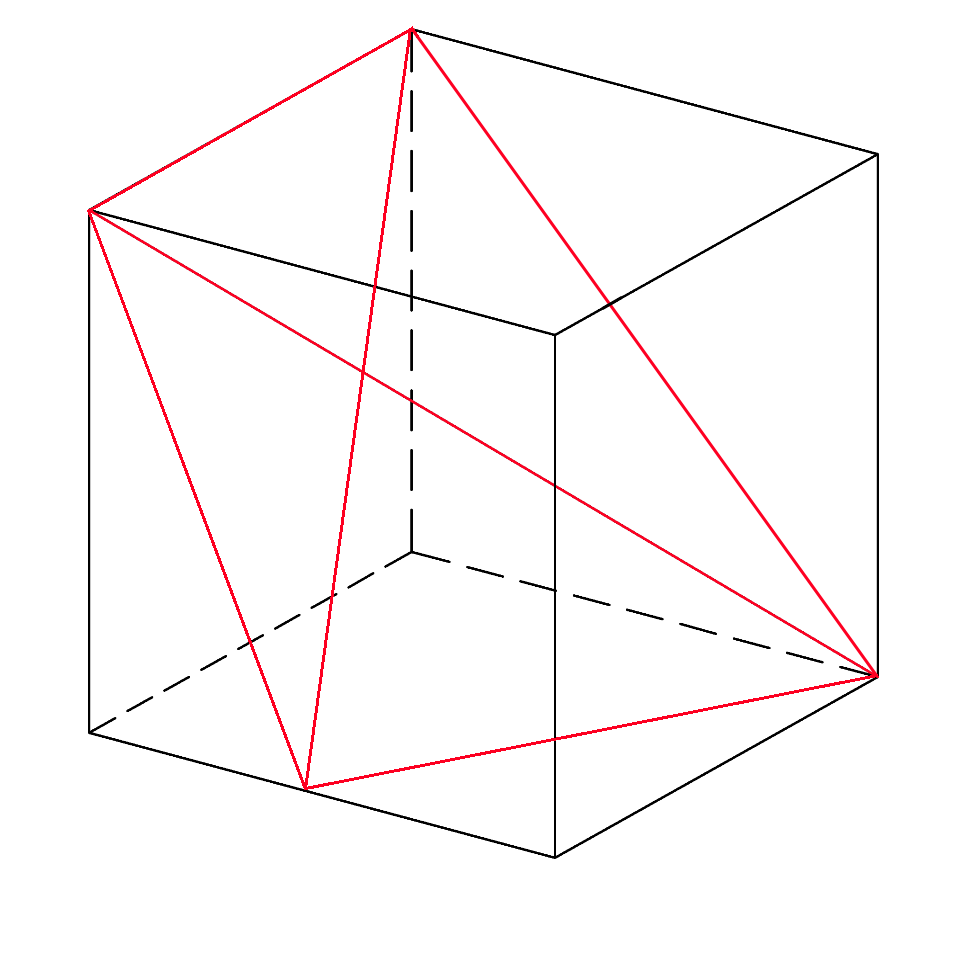
СОФИЯ

2019

**УВОД**

“Axyno” представлява смартфон приложение, предназначено за решаване на математически задачи, включващи стереометрия в тях. Целта е да се използват възможностите на мобилното устройство за симулация на виртуална заобикаляща среда, чрез която чертаенето и решаването на подобен тип задачи да могат да се пренесат в триизмерното пространство.

Проблемът, който приложението решава е невъзможността да се визуализира триизмерно тяло върху двуизмерна равнина достатъчно ясно, например чертеж върху лист хартия. Независимо чрез каква проекция представяме триизмерния образ винаги компенсираме определен аспект, което понякога прави пространственото възприемане на чертежа доста трудоемка задача.

**Пример:** В следния чертеж на вписана пирамида в куб, независимо от проекцията са нужни поне няколко секунди фокусиране и пространствено визуализиране, докато успеем да си представим изкривената форма на пирамидата.

**C1**

*Фиг 0.1 Чертеж в ортографска проекция*

**C**

**А**

**B**

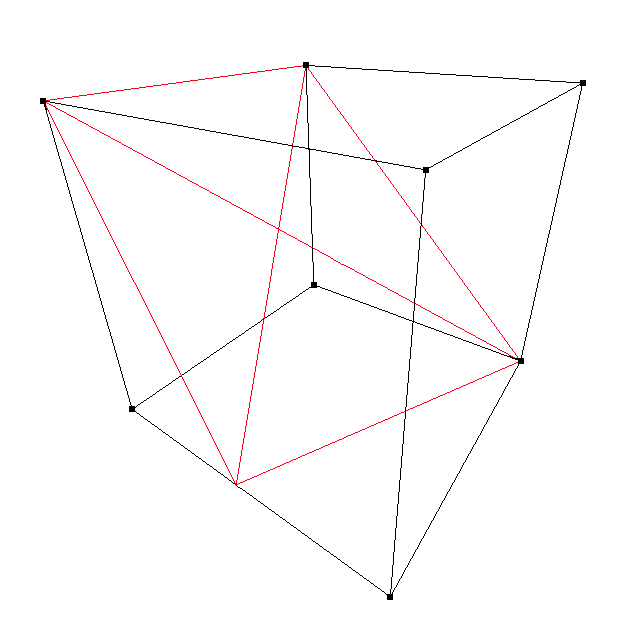
**D1**

**C1**

**D**

**А1**

**B1**



*Фиг 0.2 Чертеж в перспективна проекция*

**B**

**D1**

**B1**

**А1**

**D**

**C**

**А**

“Axyno” е единственото приложение към момента на пазара, което представлява опростен редактор на триизмерни обекти, създаден специално за използване със смартфон във виртуална среда. Приложението може да бъде използвано в училища и университети за елиминиране на проблемите при начертаването на една задача и акцентиране върху самата логическа последователност за решаването ѝ.

Работата с приложението става посредством отделен headset, предназначен за смартфони, като за целите на разработката беше използван Google Cardboard.

ПЪРВА ГЛАВА

Технологии за реализация на дипломния проект

1. **Основни принципи, технологии и развойни среди, използвани при реализацията**
   1. **Технологии**
      1. **Unreal Engine**



*Фиг 1.1 Unreal Engine лого*

Unreal Engine е един от най-разпространените публични игрови двигатели („game engines“) на пазара. Поддръжката на всички масови платформи заедно с безкомпромисното качество и вече 21 години разработка показват, защо това е един от най-популярните и често използвани игрови двигатели. Unreal Engine е изцяло написан на C++, като целия код е публично достъпен в Github.

Unreal Engine се грижи почти изцяло за рендерирането и менажирането на памет, създавайки абстракция за програмиста.

Няма преки пътища към създаването на потапящи се преживявания, които са правдоподобни за човешкия ум. VR изисква сложни сцени, представени при много високи стойности на кадрите. Тъй като Unreal Engine е предназначен за взискателни приложения като AAA игри, създаване на филми и фотореалистични сцени, той отговаря на тези изисквания и осигурява солидна основа за изграждане на продукти за всички VR платформи - от компютри през конзоли до мобилни

Главните предимства при използване на игрови двигател са, че те се грижат за:

* Рендериране
* Управление на памет
* Засичане на колизии и още много.

Като бонус Unreal Engine предоставя и невероятно лесно интегриране на Виртуална Реалност.

* + 1. **C++**

*Фиг 1.2 C++ лого*

**„**C++ или CPP („Си Плюс Плюс“) е език за програмиране с общо предназначение. Той има императивни, обектно-ориентирани и общи програмируеми функции, като същевременно осигурява възможности за манипулиране на паметта на ниско ниво.“ – Wikipedia

Unreal Engine имплементира собствен „Garbage Collector“ и ” Unreal Property System (Reflection)“ система базирана изцяло на макрота. Две функции, нужни на един модерен игрови двигател, които C++ като език известен с максимална производителност, ефективност и гъвкавост не предлага.

Системата „Garbage Collector” изцяло се грижи за управлението на памет като следи, кои обекти се използват и кои вече не се. “Modern C++” имплементира няколко „умни поинтери“ („Smart pointers”), но Unreal Engine скрива и това от програмистите като менажира цялата памет.

“Reflection” системата дава способността на една програма да се изследва по време на изпълнение. Това е изключително полезно и е основополагаща технология на Unreal Engine, много системи като детайлни панели в редактора, сериализация, „Garbage Collection”, репликация на мрежата и комуникация „Blueprint“ / C ++, разчитат на тази система. C ++ обаче не поддържа под никаква форма „Reflection” система, така че Unreal има своя собствена система за събиране, обработване и манипулиране на информация за C ++ класове, структури, функции, член променливи и енумерации.

„Reflection” системата не е автоматична. За да бъде „видим“ за нея тип или свойство на един клас трябва да анотира. „Unreal Header Tool“ ще събере тази информация, когато компилирате проекта.



*Фиг 1.3 Демонстрация на „Unreal Property System (Reflection)“*

В Unity поведението на определен игрален обект се управлява от закачени за него „компоненти“. Въпреки че има широк набор от вече готови такива може да е необходимо по-персонализирано управление. В такива случаи може да бъде създаден допълнителен собствен компонент с помощта на външен C# скрипт. Скриптът бива изпълнен от съответния интерпретатор, който го превежда от символен на машинен език, пропускайки стъпката по генериране на бинарен файл, както правят например компилаторите.

Едно от най-големите предимства на софтуерната платформа “.NET”, която Unity използва, е интегрираното автоматично управление на паметта. За това се грижи т.нар. система „garbage collector“, която е част от CLR (Common Language Runtime) компонента на платформата. Основната задача на тази система е да следи кога заделената памет за променливи и обекти вече не се използва и да я освобождава. Това освобождава програмиста от следене на използваните ресурси и проблеми като memory leaks („теч на памет“).

* 1. **Мобилни платформи**
     1. **Android**



*Фиг 1.5 Android лого*

Android („Андроид“) е операционната система на Google Inc. за мобилни устройства. Базирана е на Linux ядрото, като последната стабилна версия до момента е 9.0, именувана „Pie“. Unix-подобната операционна система е с open source (или „отворен код“) модел като повечето устройства съдържат и софтуер със затворен код.

За развитието на операционната система се грижат относително голям брой програмисти и софтуерни разработчици, които създават така наречените „Apps“ (от Applications - приложения), които представляват малки програми, целящи да разширят функциалността на системата. Тези приложения могат да бъдат сваляни от Интернет или от официалния магазин на Android, наречен Google Play, като приложенията от там могат да бъдат както безплатни, така и платени. Процеса по публикуване на приложение в магазина е доста лесен и общодостъпен – единственото условие е разработчика да има направен Google профил. Самия акаунт също е безплатен и за разлика от други мобилни магазини няма нито закупуване на лицензи, нито сертификационни тестове и периоди.

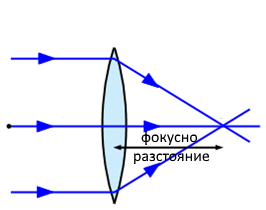
* + 1. **(iOS)**

(optional for now)

* 1. **Хардуер**
     1. **Google Cardboard**

*Фиг 1.6 Google Cardboard с поставен смартфон*

Google Cardboard е мобилна платформа за виртуална реалност, разработена от Google Inc. За разлика от други очила за виртуална реалност Google Cardboard използва процесорната мощност на смартфон за да симулира виртуална среда. Следенето на движенията и завъртанията също се осъществява от телефона, благодарение на вградените в него сензори - акселерометър и жироскоп.

 За да може образа на екрана да бъде чист и ясен продуктът използва двойно изпъкнали (биконвексни) лещи, поставени пред всяко око с фокусно разстояние от 45мм.

*Фиг 1.7 Действие на двойно изпъкнала леща*

Google предоставя три комплекта за разработка на софтуер (SDK – System Development Kit) за платформата си: един за операционната система Android, използващ Java, един за игровия двигател Unity, написан на C#, и един за операционната система iOS.

Понеже приложенията, проектирани за Google Cardboard биват пускани на мобилно устройство, след което самото то бива поставено в определеното за целта място, нямаме пряк достъп до сензорния екран на телефона. Затова интеракцията с приложението става посредством бутон, разположен в горния десен ъгъл на устройството, който притиска кондуктивно парче до екрана регистрирайки допир.



*Фиг 1.8 Местоположение на бутона за интеракция*

**ВТОРА ГЛАВА**

Проектиране на структурата на Android мобилни приложения за виртуална реалност

1. **Функционални изисквания към Android приложения**
   1. **Изисвания към изграждането на логиката**

Трябва да бъде изградена осносна структура на приложението. Да се направят основни класове чрез които добавянето на нови функционалности да става бързо и лесно. Да се изгради лесна и логически-издържана комуникация между различните части на приложението.

* 1. **Изисквания към графиката на приложението**

Приложението трябва успешно да симулира виртуална заобикаляща среда. Всеки текст, картинка и символ трябва да бъде ясно разграничаван и лесно разбираем. Графиката на приложението не трябва да забавя изпълнението на програмата, като кадрите в секунда е желателно да не падат под 30. Чертежът трябва да е с неограничена големина без това да се отразява директно на производителността.

* 1. **Потребителски интерфейс и итерация с потребителя**

Потребителският интерфейс на приложението трябва да бъде максималко опростен и разбираем. Цялата навигация ще трябва да се осъществява само чрез Google Cardboard визьора, затова всички менюта и бутони трябва да са с достатъчно голям размер че да не бъде изпитвано затруднение при използването им.

* 1. **Функционалност**

Приложението трябва да изисква възможно най-малко права от устройството. Данните, прочетени от акселерометъра и жироскопа на смартфона, трябва да бъдат извличани и обработвани достатъчно бързо, подсигурявайки следене на въртенето на главата в реално време.

* 1. **Производителност**

Приложението не трябва да забива или замръзва без причина. Моменти на насичане, прекъсване или други дефекти не трябва да бъдат забелязвани. Също така латентността трябва да бъде минимална през цялото време.

1. **Съображения за избор на програмни средства и развойната среда**
   1. **Игрови двигател Unity**

За реализацията на приложението беше взет избора да се използва игрови двигател поради многото си предимства при разработка, като например:

* Повечето от сложните и времеемки аспекти от приложението като управление на паметта, зареждане на асети, изчисляване на осветлението и др. са вече написани и готови за ползване;
* Повечето игрални двигатели са междуплатформени, което прави експортирането на приложение за специфична платформа доста лесна задача

Игровият двигател, който беше избран е Unity. Причината е че инструментите му не са непознати, тъй като вече различни проекти са били правени ползвайки него както за училище, така и за хоби. Това пропуска стъпката от заучаване на нови технологии и увеличава продуктивността.

* 1. **Google Cardboard платформа**

Понеже приложението е проектирано като продукт, имплементиращ виртуална реалност, с намерение да се използва предимно в училища използването на смартфон беше очевидния избор за устройство. Google Cardboard платформата допринася за лесната портативност и общодостъпността на приложението. Тя е евтина и лесна за работа, като не се изисква по-специализиран хадруер за нея.

* 1. **Операционна система Android**

След обстойно проучване на пазара за мобилни операционни системи се стигна до заключението че Android е най-уместният избор поради следните причини:

* Наличие на физически устройства, на които да бъде тествано приложението;
* Не се изискват допълнителни финансови средства;
* Липса на опит с други платформи;
* Лесно и бързо публикуване в магазина за приложения Google Play;
  1.  **GitHub система за контрол и управление на версиите**

*Фиг 2.1 GitHub лого*

След възникнали проблеми в системата за контрол на версиите на Unity, наречена Unity Collaborate, преместването към GitHub беше наложително. GitHub представлява уеб-базирана услуга за качване на софтуерни проекти върху отдалечен сървър (т.нар. хранилище) и осъществяването на съвместни разработки бърху тях. Базиран е на Git системите за контрол и е избран заради вече съществуващия опит с него.

* 1.  **Trello мениджър на задачи**

*Фиг 2.2 Trello лого*

Trello е уеб-базирано приложение за управление на проекти и отделните задачи по тях. Използването му се състои от класифициране на подзадачите спрямо различни критерии и следене на всичко планирано до момента, както и визуализиране на прогреса при разработката.

* 1. **Notepad++ текстов редактор**



*Фиг 2.3 Notepad++ лого*

Цялостната процедура по писане на кода беше осъществена в Notepad++. То представлява опростен текстов редактор, разпространяващ се като софтуер с отворен код. Главното му предимство е малкия му размер – почти 23MB след инсталация, както и опростения му интерфейс и оцветяване на синтаксиса за над 50 програмни езика.

* 1.  **Paint.net графичен редактор**

*Фиг 2.4 Paint.net лого*

Paint.net е безплатен редактор на растерна графика, разработен на .NET платформата. От функционална гледна точка той предоставя много повече инструменти от редактора на изображения по подразбиране в Windows – “Paint“. Това, заедно с малкия му размер и високата производителност, го прави изключително добър за работа с малки и средни проекти.

**(blender if used at all when design comes to place)**

**ТРЕТА ГЛАВА**

Програмна реализация на “Axyno”

1. **Архитектура на приложението**
   1. **Типове обекти за построяване**

Чертежът бива построен чрез три основни обекта, свързващи се помежду си. Той може да съдържа неограничен брой съставляващи го части.

* + 1. **Точка**

Точката представлява единична позиция в триизмерното пространство. Тя съдържа информация за съответните си координати по трите оси и в приложението бива репрезентирана като сфера.

Използва се за да обуславя определена линия. Към една точка могат да бъдат свързани безброй много прави.

* + 1. **Линия**

Линията е права, ограничена от две точки. Тя съдържа информация за двете си точки и бива визуализирана като дълъг цилиндър, свързващ ги.

Линията репрезентира определен ръб или права в чертежа, като между две точки може да съществува само една линия.

* + 1. **Ъгъл**

Ъгълът е геометричен обект, съставен от две прави с обща начална точка, наричана връх на ъгъла. Съхранява инфо за двете линии, съставляващи го и в приложението се представя като дъга от едната линия до другата.

Обектът се използва за да се визуализира числената големина на ъгъла, отразяващ степента на завъртане на едната линия около върха, така че тя да съвпадне с другата.

* 1. **Разделение на класовете**

Всеки инструмент или действие в програмата си има отделен клас, определящ функциалността им.

|  |  |
| --- | --- |
| Инструмент/Действие | Съответстващ клас |
| Инструменти | ToolButton.cs |
| Построяване на линия | CreateLine.cs |
| Смяна на цвета | ChangeColor.cs |
| Смяна на типа на линията | ChangeLineType.cs |
| Делене на линия | SubdivideLine.cs |
| Построяване на перпендикуляр | CreatePerpendicular.cs |
| Построяване на ъгъл | CreateAngle.cs |
| Разширение на линия | ExpandLine.cs |
| Изтриване на обект | Delete.cs |
| Избутване на фигура (и сливането ѝ) | ExtrudeFace.cs |

* 1. **Главни класове на приложението**
     1. **Tool.cs**

Основният клас за всички действия с обекти е Tool.cs

* + 1. **CreatedObject.cs**
  1. **Карта на всички класове**
  2. **Интеграция с Cardboard SDK**

1. **Графика на приложението**
   1. **Потребителски интерфейс**
   2. **Графично представяне на елементите (shaders n such)**
2. **Реализация на инструментите и действията**
   1. **On-screen buttons**
      1. **Отваряне на меню**
      2. **Избор на инструмент**
      3. **Завъртане на чертежа**
      4. **Клавиатура (when ready)**
   2. **Инструменти**
      1. **Select Tool**
      2. **Rotate Tool**
   3. **Действия**
      1. **Build Line / Постояване на линия**
      2. **Change Color / Смяна на цвета**
      3. **Change Line type / Смяна на типа линия**
      4. **Subdivide / Деление на линия**
      5. **Build Perpendicular / Строене на перпендикуляр**
      6. **Create Angle / Построяване на ъгъл**
      7. **Expand Line / Разширение на линия**
      8. **Delete Object / Изтриване на обект**
      9. **Extrude (and merge to a point) / Избутване на форма (и сливането ѝ в точка)**

**ЧЕТВЪРТА ГЛАВА**

Ръководство на потребителя

1. **Инсталиране**

Инсталирането може да се осъществи по два начина: Първият начин е през Google Play приложението или уебсайта, а втория – чрез файл с разширение apk.

* 1. **През Google Play**

След влизане в Google Play приложението, което е инсталирано на всеки телефон с операционна система Android, се търси приложението по име или по производител. След откриването му може да бъде видяно описание на приложението, снимки и последни актуализации.

При натискане на бутона за инсталиране от страницата, приложението ще започне да се сваля, а след това и ще бъде инсталирано автоматично от системата.

Ако инсталирането трябва да бъде осъществено от уеб портала на Google Play, то е необходимо да се намери страницата на приложението по същия начин и да се натисне бутона за инсталация (трябва да е влязъл резистриран профил). Веднага щом телефонът получи достъп до интернет, приложението ще започне да се сваля и ще бъде инсталирано отново от системата.

* 1. **Чрез apk файл**

За да бъде инсталирано от apk файл, най-напред е необходимо да бъде разрешено инсталирането на неоторизирани приложения от:

Settings > Security & Privacy > Install unknown apps

След това apk файла трябва да бъде качен на мобилното устройство посредством USB, Bluetooth или по друг начин. Откриването на прехвърления файл става посредством файлов мениджър като след стартирането му системата ще попита за потвърждение дали приложението да бъде инсталирано. След преглед на исканите права и разрешаване на инсталацията приложението ще бъде инсталирано, като след това иконата му ще с епояви в менюто.

1. **Стартиране**
2. **Използване**
   1. **Select**
   2. **Ц.**
   3. **Етц.**
3. **Деинсталиране**

Премахването на приложението става по стандартния за Android платформата начин – от менюто „Приложения“ в настройките на устройството.

1. **Поддръжка**
   1. **Актуализации**

Актуализациите (или ъпдейтите) се предоставят от Google Play приложението в момента в който такава е налична и телефона има интернет връзка

* 1. **Докладване за грешки**

Предоставя се от Android платформата. Когато приложението неочаквано се изключи или блокира поради някаква причина, ще се появи диалогов прозорец, в който може да се избере да се докладва грешката.