**СОДЕРЖАНИЕ**

[Введение 4](#_Toc53442214)

[1 Аналитический обзор технических средств 5](#_Toc53442215)

[1.1 Обзор игр «Танки» 5](#_Toc53442216)

[1.2 Обзор графических библиотек *OpenGL* и *DirectX* 7](#_Toc53442217)

[1.3 Сравнение графических средств *OpenGL* и *DirectX* 11](#_Toc53442218)

[1.4 Обзор технологии *Windows* *Sockets* 14](#_Toc53442219)

[2 Архитектура и программная реализация игрового приложения 15](#_Toc53442220)

[2.1 Определение основных этапов в разработке игрового приложения 15](#_Toc53442221)

[2.2 Программная реализация игрового приложения 15](#_Toc53442222)

[3 Верификация и анализ результатов 16](#_Toc53442223)

[3.1 Эксплуатация *2D* игры «Танки» 16](#_Toc53442224)

[3.2 Верификация работы приложения 16](#_Toc53442225)

[3.3 Модульное тестирование 16](#_Toc53442226)

[Заключение 17](#_Toc53442227)

[Список используемых источников 18](#_Toc53442228)

# **ВВЕДЕНИЕ**

В настоящее время особенно широкое распространение получили различные компьютерные игры. Они привлекают обширную аудиторию, в которую входят представители всех возрастов и слоев населения. Наблюдается стремительное развитие игрового программного обеспечения, постоянное усовершенствование методов реализации игрового процесса, графических представлений, разработка новых методов взаимодействию с приложением. Они оказали столь существенное влияние на общество, что в информационных технологиях отмечена устойчивая тенденция к геймификации для неигрового прикладного программного обеспечения. Создание компьютерной игры является способом выражения своих идей и создания своего уникального проекта.

Целью курсового проекта является разработка игрового приложения *Windows Form* «Танки» для игры на одном экране двух пользователей друг против друга. Выполнение курсового проекта потребовало решения следующих задач:

– выполнить аналитический обзор игр жанра «Танки»;

– разработать основные классы;

– использовать шаблоны проектирования «фабричный метод» и «декоратор»;

– разработать программный код игры;

– провести тестирование разработанного приложения.

Данную игру предлагается реализовать на языке *C#* c использованием графики *OpenGL*. Язык *C#* основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода*.*

# **1 АНАЛИТИЧЕСКИЙ ОБЗОР ТЕХНИЧЕСКИХ СРЕДСТВ**

## **1.1 Обзор игр «Танки»**

В игре «Танки» принимают участие два игрока, каждый из которых управляет своим танком с помощью клавиатуры. В начале игры танки имеют одинаковые характеристики. Танки перемещаются по игровому полю с неоднородной территорией. Пули из пулемёта достигают цели моментально, у снарядов скорость ограниченна, так что танки могут уклоняться от выстрелов. Победителем считается игрок, уничтоживший танк соперника.

Далее жанр будет рассмотрен более подробно на примере нескольких аналогичных игр.

Одна из самых известных игр жанра «Танки» – *Battle City* (рисунок 1.1), выпущенная в 1985 году фирмой *Namco*.



Рисунок 1.1 – Геймплей игры *Battle City*

Игровой полигон действий в игре представлен в виде сверху. Игрок управляет своим танком и должен уничтожить все вражеские танки на уровне, которые постепенно появляются на игровом поле. Вражеские танки пытаются уничтожить штаб игрока, который представлен внизу игрового поля в виде орла, и его танк. На каждом уровне нужно уничтожить 20 вражеских танков. Если противник сможет разрушить штаб или лишит игрока всех жизней – игра будет окончена.

Игра состоит из 35 основных уровней. Игровое поле представляет собой квадратное поле из 169 секторов (13×13). После прохождения 35 уровней они начинаются заново, но становятся сложнее из-за того, что изменяется набор вражеских танков (набор 30-го уровня), а также из-за более быстрого респауна врагов.

Каждая карта содержит различные типы местности и препятствий:

– кирпичные стены, которые могут быть разрушены от одного или нескольких выстрелов танка;

– стальные стены, которые игрок может разрушить, если он собрал три или более усиленных звезд;

– кусты, снижающие видимость танков и снарядов;

– лёд, который затрудняет управляемость движения танка;

– вода, блокирующая передвижение, но пропускающая снаряды.

Игра содержит четыре типа танков противника, которые различаются скоростью и прочностью. На более поздних уровнях игра становится сложнее, поскольку вражеские танки могут действовать как приманка, чтобы отвлечь игроков от своей базы, чтобы другой танк мог ее уничтожить. Кроме того, мигающие танки могут быть уничтожены для бонусов.

*Tank Battalion* (рисунок 1.2) –игра, выпущенная фирмой *Namco* в 1980*.* Игрок, управляющий танком, должен уничтожать 20 вражеских танков на каждом уровне. Вражеские танки пытаются уничтожить базу игрока (изображенную на карте как орел), а также сам танк игрока. Уровень завершается, когда игрок уничтожает все 20 вражеских танков, но игра заканчивается, если база игрока уничтожена или у него кончились жизни.



Рисунок 1.2 – Геймплей игры *Tank Battalion*

В 1990 году компанией *Namco* была выпущена игра *Tank Force* (рисунок 1.3),сиквелигры *Battle City*.



Рисунок 1.3 – Геймплей игры *Tank Force*

Геймплей очень похож на *Tank Battalion*, за исключением того, что на этот раз одновременно могут играть до двух (на вертикальной модели) или четырех (на коктейле) игроков, появились семь новых («обычных») типов вражеских танков. В игре есть 15 типов бонусов, которые позволяют игрокам увеличить огневую мощь своих танков и повысить свой счет. В отличии от *Tank Battalion* и *Battle City*, каждый уровень имеет размер 17x13 блоков вместо 13x13. Игроки не могут разрушить свои собственные стены штаб-квартиры, поэтому, когда один игрок стреляет в другого, они будут отталкивается назад. У игры есть окончание, которое будет видно после прохождения всех тридцати шести раундов.

## **1.2 Обзор графических библиотек *OpenGL* и *DirectX***

Для решения поставленной задачи существуют графические библиотеки *OpenGL* и *DirectX*. Далее будет рассмотрена характеристика каждой библиотеки и проведен аналитический обзор.

*OpenGL* (*Open Graphics Library* – открытая графическая библиотека, графическое *API*) – это программный интерфейс, применяемый для получения программно-генерируемых изображений и приложений, использующих двух- или трёхмерные объекты, а также растровые изображения [1].

*OpenGL* появился в результате переработки специализированного *3D*-интерфейса этой фирмы *IrisGL* на универсальный, независимый от платформы. Данный интерфейс позволяет создавать трехмерные графические изображения независимо от используемых оконных интерфейсов, операционной системы и графических устройств.

В основе *OpenGL* лежит библиотека графических функций, предоставляющих разработчику все необходимое для создания и отображения плоских (*2D*) и пространственных (*3D*) моделей и анимации. Эта библиотека, содержащая более 250 функций, обеспечивает средства для управления графическими примитивами, выполнения матричных преобразований, задания источников света, теней, управления цветам.

Основной принцип работы *OpenGL* – получение наборов векторных графических примитивов в виде точек, линий и многоугольников с последующей математической обработкой полученных данных и построением растровой картинки на экране и/или в памяти. Векторные трансформации и растеризация выполняются графическим конвейером, который по сути представляет собой дискретный автомат. Пример рендеринга треугольника в *OpenGL* представлен на рисунке 1.4.

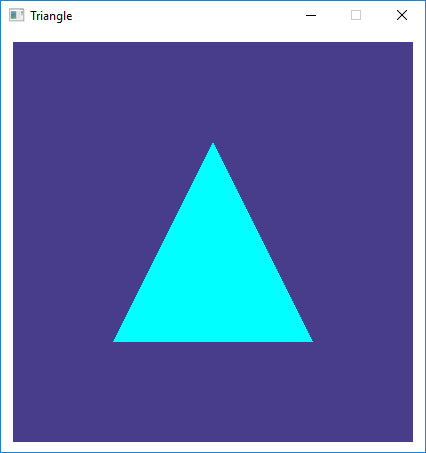


Рисунок 1.4 – Пример рендеринга треугольника в *OpenGL*

*OpenGL* является низкоуровневым процедурным *API*, что вынуждает программиста диктовать точную последовательность шагов, чтобы построить результирующую растровую графику (императивный подход). Это является основным отличием от дескрипторных подходов, когда вся сцена передается в виде структуры данных, которая обрабатывается и строится на экране. С одной стороны, императивный подход требует от программиста глубокого знания законов трёхмерной графики и математических моделей, с другой стороны – даёт свободу внедрения различных инноваций.

*OpenGL* ориентируется на следующие две задачи:

* скрыть сложности адаптации различных 3D-ускорителей*,* предоставляя разработчику единый *API*;
* скрыть различия в возможностях аппаратных платформ, требуя реализации недостающей функциональности с помощью программной эмуляции.

Для подтверждения независимости от языка программирования были разработаны различные варианты привязки (*binding*) функций *OpenGL* или полностью перенесены на другие языки. Например, библиотека *Java 3D*, которая может использовать аппаратное ускорение *OpenGL*. Прямая привязка функций *OpenGL* для *Java* реализована в *Lightweight Java Game Library*. Имеются также варианты привязки *OpenGL* для языков *C++* и *C#*. Для связи *OpenGL* c *C#* используются библиотеки *Tao Framework, OpenTK и SharpGL.*

Основные функции и возможности *OpenGL*:

* геометрические примитивы позволяют задавать математическое описание объектов;
* цветное кодирование в формате RGBA или в режиме индексированных цветов;
* просмотр и моделирование позволяют размещать трехмерные объекты на сцене, передвигать камеры вокруг сцены и выбирать точку обзора, для которого выполняется рендеринг;
* текстуры помогают привнести реализм в модели за счет имитации реальных поверхностей (их изображения накладываются поверх «скелета» модели, составленного из многоугольников);
* освещение, которое задаётся функцией описания свойства материала и источников света для сцены;
* двойная буферизация позволяет устранить мерцание при анимации;
* антиалиасинг сглаживает «ступенчатые» края цветовых переходов путём изменения цвета и интенсивности точек вблизи линии резкого перехода;
* затенение по методу Гуро закрашивает *3D*-объектов, при которой сначала рассчитывается освещение вдоль ребер треугольников на основании данных об освещенности вершин, а затем на основании этих данных и направления градиентов изменения цвета закрашивается внутренняя поверхность треугольников.
* *Z*-буферизация используется для сохранения информации о «глубине» трехмерного объекта (*Z*-координаты);
* атмосферные эффекты туман или дымка делают компьютерные изображения более реалистичными;
* альфа-смешение использует значение Альфа-буфера (информации о прозрачности пикселов) как составляющую расширенного цветового кода растрового изображения в формате *RGBA*, позволяя комбинировать цвет обрабатываемого фрагмента с цветом точек, которые уже хранятся в буфере изображения;
* трафаретные планы ограничивают отрисовку на некоторых областях экрана по маске (трафарету);
* cписки отображения позволяют сохранять команды отрисовки в некотором списке для дальнейшего рендеринга;
* полиномные вычисления используются для поддержки *NURBS*-сплайнов, c помощью которых можно рисовать плавные кривые через небольшой набор опорных точек, что исключает необходимость сохранять все промежуточные значения.
* обратная связь, выделение и выбор – функции которые дают возможность создавать приложения, позволяющие пользователю выбирать область экрана или отдельный объект, изображенный на экране;
* растровые примитивы: растровые изображения (прямоугольные картинки, состоящие из точек);
* операции с пикселами;
* различные преобразования – вращение, масштабирование, перемещение, перспективные искажения и т.д. [2]

Библиотека *OpenGL* является одной из самых перспективных библиотек трехмерной графики, имеющихся в распоряжении разработчиков. Основное преимущество данной библиотеки – аппаратная и платформерная независимость, позволяющая экономить силы разработчиков при создании приложений, работающих на различных платформах.

Следующее графическое средство, на которое будет сделан обзор – *DirectX*.

*DirectX* – это набор *API*, разработанных для решения задач, связанных с программированием под *Microsoft Windows*. Наиболее широко используется при написании компьютерных игр. Пакет средств разработки *DirectX* под *Microsoft* *Windows* бесплатно доступен на сайте *Microsoft*. Зачастую обновлённые версии *DirectX* поставляются вместе с игровыми приложениями [3].

Сам *DirectX* состоит из следующих модулей:

* *DirectX Graphics*: набор интерфейсов, ранее (до версии 8.0) делившихся на *DirectDraw*:интерфейс вывода растровой графики, и *Direct3D*:интерфейс вывода трёхмерных примитивов;
* *DirectInput*: интерфейс, используемый для обработки данных, поступающих с клавиатуры, мыши, джойстика и прочих игровых контроллеров;
* *DirectPlay*: интерфейс сетевой коммуникации игр;
* *DirectSound*: интерфейс низкоуровневой работы со звуком;
* [*DirectMusic*](https://ru.wikipedia.org/w/index.php?title=DirectMusic&action=edit&redlink=1): интерфейс воспроизведения музыки в форматах *Microsoft*;
* *DirectShow*: интерфейс, используемый для ввода/вывода аудио и/или видео данных;
* *DirectSetup*: часть, ответственная за установку *DirectX*;
* *DirectX* *Media Objects*: реализует функциональную поддержку потоковых объектов;
* *Direct2D*: интерфейс вывода двухмерной графики.

Пример рендеринга куба в *DirectX* представлен на рисунке 1.5.

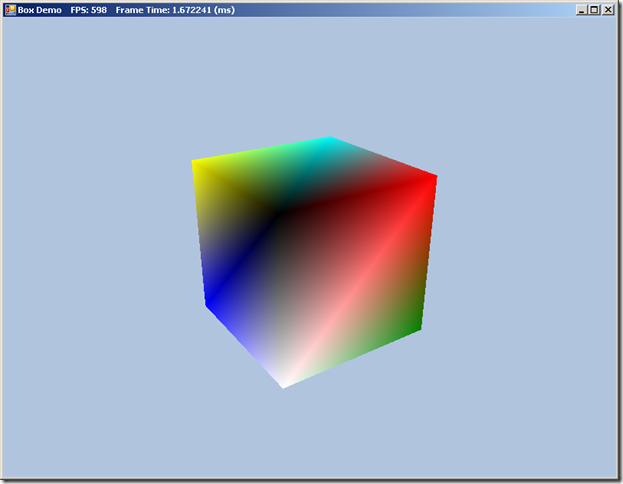


Рисунок 1.5 – Рендеринга куба в *DirectX*

*DirectX* появился именно в помощь разработчикам компьютерных игр и мультимедийных приложений и позволяет создавать программы с очень реалистичной графикой и звуками. Поскольку *DirectX* «заточен» под игры, он имеет инструменты, позволяющие обрабатывать данные, получаемые с устройств управления – с клавиатуры, мыши или игрового джойстика. Изначально нацеленный на разработку видеоигр, *DirectX* стал популярен и в других областях разработки программного обеспечения. К примеру, *DirectX*, наряду с [*OpenGL*](https://ru.wikipedia.org/wiki/OpenGL), получил очень широкое распространение в инженерном/математическом ПО.

## **1.3 Сравнение графических средств *OpenGL* и *DirectX***

Для выбора графической библиотеки к поставленной задаче будет проведено сравнение библиотек *OpenGL* и *DirectX*. Каждая из этих библиотек имеет свои особенности, которые будут рассмотрены более подробно в данном разделе.

Библиотека *OpenGL (Open Graphics Library)* считается одной из лучших библиотек, как для профессионального применения, так и для игр. Однако, у неё существуют конкуренты.

Главных конкурентом считается *Direct3D* из пакета *DirectX*, разработанный компанией *Microsoft*.

Основной особенностью *OpenGL* можно считать простоту. Процесс обработки примитивов контролирует ядро *OpenGL*. Для передачи данных используется процедурная модель. В каждый момент времени состояние *OpenGL* определяется через набор переменных, задающих параметры обработки. Каждый новый переданный треугольник проходит обработку в соответствии с текущим состоянием. Такой механизм весьма эффективен, а код обычно короток и прост, поэтому *OpenGL* подходит для начала изучения трехмерной графики.

Также, преимуществом *OpenGL* является кроссплатформенность – переносимость программ с одной платформы на другую. Благодаря ей программы, написанные с помощью *OpenGL* можно успешно перенести на такие платформы как *Unix*, *Linux*, *SunOS*, *IRIX*, *Windows*, *MacOS*. В свою очередь *Direct3D* может работать только *на Intel* платформах под управлением операционной системы *Windows*.

Однако, в плане объектно-ориентированного подхода *OpenGL* уступает *Direct3D*. *OpenGL* работает по принципу конечного автомата, переходя из одного состояния в другое, совершая при этом какие-то преобразования.

Ещё одним преимуществом *Direct3D* является поддержка дешёвого оборудования, *OpenGL* же поддерживается не на всех графических картах, но для профессиональных ускорителей *OpenGL* является стандартом де-факто.

Структура *DirectX* значительно отличается от *OpenGL*. *DirectX* основан на модели *COM* (*Component Object Model*). Это означает, что в отличие от простого вызова функций эта модель предполагает выполнение некоторых дополнительных действий, связанных с компонентной архитектурой *DirectX*. Недостатком такой архитектуры является использование большого объема кода, необходимого для написания простой программы на *DirectX*. Для упрощения программирования разработчики *Microsoft* создали отдельную библиотеку *DirectX Common Files*, которая скрывает часто используемый код [4].

Архитектура *DirectX* сильно отличается от *OpenGL*, но в их развитии все более заметны тенденции к сближению. Такая ситуация возникает прежде всего потому, что обе библиотеки предназначены для эффективной работы с аппаратурой, и чем ближе их структура будет к "железу", тем меньше времени будет уходить на преобразование команд пользователя в команды аппаратуры.

По поддержке аппаратных функций *OpenGL* и *DirectX* эквиваленты. Новые функции в *OpenGL* доступны через механизм расширений, а в *DirectX* они появляются в новых версиях.

Важным достоинством *OpenGL* является открытый стандарт, разрабатываемый некоммерческой организацией *Khronos* *Group* при участии сообщества. Большинство крупных производителей *GPU* (*nVidia, AMD, Intel*), так или иначе, влияли на *OpenGL*. Что касается *DirectX*, то здесь ситуация прямо противоположная. Только *Microsoft* может вносить какие-либо изменения в библиотеку. Таким образом, именно *Microsoft* в конечном итоге определяет все пути развития библиотеки, и если путь был выбран неверно, это может быть исправлено только в новой версии.

Между *OpenGL* и *DirectX* существуют различия в том, как каждый *API* обрабатывает объединение вершин для формирования примитивов, которые изложены ниже в таблице 1.1.

Таблица 1.1 – Сравнение характеристик *DirectX* и *OpenGL*

|  |  |  |
| --- | --- | --- |
| Характеристики | *OpenGL* | *DirectX* |
| Смешивание вершин | Отсутствует | Имеется |
| Множество операционных систем | Имеется | Отсутствует |
| Механизм расширения | Имеется | Имеется |
| Разработка | Многопользовательская разработка | *Microsoft* |
| Точная спецификация | Имеется | Отсутствует |
| Двухсторонне освещение | Имеется | Отсутствует |
| Объёмные текстуры | Имеется | Отсутствует |
| Аппаратно-независимые *Z*-буферы | Имеется | Отсутствует |
| Накопительные буферы | Имеется | Отсутствует |
| Полноэкранное сглаживание | Имеется | Имеется |
| Размытость | Имеется | Имеется |
| Глубина резкости | Имеется | Имеется |
| Стерео-рендеринг | Имеется | Отсутствует |
| Атрибуты размера | Имеется | Отсутствует |
| Выборка | Имеется | Отсутствует |
| Параметрические кривые и поверхности | Имеется | Отсутствует |
| Геометрия кэша | *Display Lists* (Списки отображения) | *Vertex Buffers* (Буфер вершин) |
| Эмуляция системы | Аппаратные средства отсутствуют | Определяет приложение |
| Интерфейс | *Procedure calls* (Процедурные вызовы) | *COM* |
| Обновления | Ежегодно | Ежегодно |
| Исходный код | *Sample* (Выборка) | *SDK Implementation*  (Осуществляется *SDK*) |

Исходя из данных таблицы 1.1, можно сделать вывод, что *OpenGL* подойдёт для тех разработчиков, для которых важна кроссплатформенность, обеспечивающая доступность всех новых функций и на *Windows XP*, где *Direct3D* 10/11 отсутствует. Можно отметить, что драйвера *Windows* не имеют поддержки *OpenGL,* поэтому их нужно ставить с сайта производителя, и такая вещь как *SDK* в *OpenGL* отсутствует. Кроме того, *OpenGL* не подходит для работы со звуком, вводом, сетью, поэтому нужно искать соответствующие инструменты, что может отразиться на времени разработки.

## **1.4 Обзор технологии *Windows* *Sockets***

*Windows* *Sockets* (*Winsock*) представляет собой интерфейс программирования (*API*), который определяет, как сетевое программное обеспечение *Windows* должно получать доступ к сетевым службам. Он определяет стандартный интерфейс между клиентским приложением *Windows* *TCP*/*IP* (например, *FTP*-клиентом или веб-браузером) и базовым стеком протоколов *TCP*/*IP*.

Библиотека *Winsock* поддерживает два вида сокетов:

– синхронные (блокируемые), которые задерживают управление на время выполнения операции;

– асинхронные (неблокируемые), которые возвращают его немедленно, продолжая выполнение в фоновом режиме, и, закончив работу, уведомляют об этом вызывающий код.

Сокеты позволяют работать со множеством протоколов и являются удобным средством межпроцессорного взаимодействия.

Независимо от вида, сокеты делятся на два типа:

– потоковые – работают с установкой соединения, обеспечивая надежную идентификацию обоих сторон и гарантируют целостность и успешность доставки данных, опираются на протокол *UDP*;

– дейтаграммные – работают без установки соединения и не обеспечивают ни идентификации отправителя, ни контроля успешности доставки данных, зато они заметно быстрее потоковых, опираются на протокол *TCP* [5].

Выбор того или иного типа сокетов определяется транспортным протоколом на котором работает сервер, – клиент не может по своему желанию установить с дейтаграммным сервером потоковое соединение.

*Windows Sockets* 2 предназначен для использования программистами на *C*/*C*++. *Windows Sockets* 2 можно использовать на всех платформах *Windows*.

В данной главе были изучены аналогичные игры, исследованы материалы о графических средствах *DirectX* и *OpenGL,* рассмотрен *WinSock API.* Проанализировав данные о этих библиотеках, можно сделать вывод, что для реализации графики в игре «Танки» следует использовать *OpenGL*, так как данная библиотека обеспечивает кроссплатформенность, благодаря которой программа будет свободно переносится на большинстве платформ, а для организации сетевых коммуникаций должен использоваться интерфейс *Windows* *Sockets.*

# **2 АРХИТЕКТУРА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ**

## **2.1 Определение основных этапов в разработке игрового приложения**

## **2.2 Программная реализация игрового приложения**

# **3 ВЕРИФИКАЦИЯ И АНАЛИЗ РЕЗУЛЬТАТОВ**

## **3.1 Эксплуатация *2D* игры «Танки»**

## **3.2 Верификация работы приложения**

## **3.3 Модульное тестирование**

**ЗАКЛЮЧЕНИЕ**

При выполнении задания первоначально был проведен аналитический обзор средств решения задачи в области разработки игровых приложений. Были изучены характеристики *OpenGL* и *C#*. Также было выявлено, что успешная игра жанра «Танки» должна быть реализована с помощью библиотеки *OpenGL* и *Windows Form*.

В качестве средства технической реализации приложения была выбрана среда разработки *Visual Studio* 2019.

При решении поставленной задачи было разработано игровое приложение, с использованием средства языка программирования *C# Windows Form* и графики *OpenGl*, позволяющее двум пользователем при помощи клавиатуры управлять двумя игроками для перемещения и осуществления выстрелов.

В соответствии с заданием была разработана *2D*-игра жанра «Танки». В качестве технического средства реализации приложения был выбран язык программирования *C*#. Игра имеет одно окно *Windows Form*, на котором друг напротив друга отображаются два игрока и несколько разновидностей препятствий. Также на нём идёт подсчёт характеристик и статусов игроков. Игроки в начале игры имеют одинаковые характеристики: броня, снаряды и боеприпасы. При столкновении игроков друг с другом, а также с препятствиями, использованы основные принципы коллизии. В игре реализована анимация и графика с использованием *OpenGl*. Также реализованы шаблоны проектирования: «фабричный метод» – для реализации генерации игровых объектов и «декоратор» – для изменения характеристик игроков.

Данная игра рассчитана на широкий круг пользователей, как на взрослых, так и на детей. Её достоинствами являются простота в управлении и низких системных требованиях. Игра способствует развитию реакции у детей, учит рассуждать и логически мыслить, развивает образное и пространственное мышление. Игра позволяет учиться быстрее реагировать на неожиданные повороты событий, что поможет в учёбе и в жизни. Также развивает концентрацию и внимание и значительно улучшает память, позволяя запоминать всё большие объёмы информации.

При выполнении курсового проекта были изучены различные свойства реализации, создания и проектирования компьютерных игр с использованием языка программирования *C#*. Также изучены основные свойства визуальных компонентов выбранной среды реализации для графического вывода компьютерной игры. Можно сделать выводы о том, что данный курсовой проект был выполнен в полном соответствии поставленным задачам.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Программирование с использованием OpenGL [Электронный ресурс] – Режим доступа: http://www.opengl.org.ru/ – Дата доступа: 28.09.2020.
2. Знакомство с OpenGL [Электронный ресурс] – Режим доступа: https://compress.ru/article.aspx/ – Дата доступа: 28.09.2020.
3. DirectX Graphics and Gaming [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/en-us/windows/win32/directx/ – Дата доступа: 30.09.2020.
4. Сравнение OpenGL и Direct3D [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/79257/ – Дата доступа: 30.09.2020.
5. Using Winsock [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/windows/win32/winsock/using-winsock/ – Дата доступа: 30.09.2020.