# 1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОГРАММНЫХ СРЕДСТВ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ

## 1.1 Технология *DirectX*

Первая версия данной технологии появилась в 1995 году, будучи включенной в пакет второго обновления *Windows95*.

Под технологией *DirectX* понимается библиотека (набор классов, функций и структур), необходимая для выполнения различных вычислительных задач, возникающих на этапе разработки мультимедийных приложений и компьютерных игр [1].

Среди задач, которые возникают на этапе разработки мультимедиа приложений можно выделить:

– загрузка трехмерных моделей;

– анимация трехмерных моделей;

– наложение текстур на объемные объекты;

– воспроизведение звуковых файлов.

На данный момент наиболее современной считается 12-я версия *DirectX*, которая вышла в 2015 году. Данная версия программного обеспечения привязана к операционной системе из линейки *Windows.*

Среди новообретенных возможностей и нововведений можно выделить следующие аспекты:

1. поддержка новых графических ускорителей;
2. более эффективное использование мощностей процессоров;
3. увеличение производительности;
4. возможность создания более сложных сцен и эффектов;
5. возможность объединения в единую графическую подсистему графических карт от различных производителей;
6. новые режимы наложения и растеризации, улучающие процесс отсечения объектов и определения столкновений;
7. трассировка лучей в реальном времени.

### **1.1.1** Microsoft Component Object Model

При программировании *DirectX* разработчику необходимо прибегнуть к объектно-ориентированной модели программирования *Microsoft* *Component Object Model* (*COM*). Данная модель не ограничивается простым вызовом функции из библиотеки, а подразумевает выполнение дополнительных действий, которые связаны с компонентной архитектурой *DirectX* [2].

*COM*-объект – это, по сути, инкапсулированный компонент функциональности, который может использоваться приложениями для выполнения одной или нескольких задач. Для развертывания один или несколько *COM*-компонентов упаковываются в двоичный файл, называемый *COM*-сервером (чаще всего это *DLL*).

Dynamic Link Library (*DLL*) – динамическая подключаемая библиотека в операционной системе *Windows*. Динамические библиотеки позволяют сделать архитектуру более модульной, уменьшить количество используемых ресурсов и упрощают модификацию системы [3].

Традиционная библиотека *DLL* экспортирует бесплатные функции. *COM*-сервер может сделать то же самое. Но *СОМ*-компоненты внутри *COM*-сервера предоставляют *СОМ*-интерфейсы и методы-члены, принадлежащие к этим интерфейсам. Приложение пользователя создает экземпляры компонентов *COM*, извлекает интерфейсы из них и вызывает методы на этих интерфейсах, чтобы извлечь выгоду из возможностей, реализованных в компонентах *COM* [4].

### **1.1.2** Основные компоненты DirectX

Технология *DirectX* состоит из следующих основных компонентов:

1. *Direct3D*;
2. *XACT3*;
3. *DXGI*;
4. *DirectCompute*;
5. *DirectX Raytracing*;
6. *DirectStorage*.

*Direct3D –* представляет собой программный интерфейс, являющийся частью графических приложений *Windows*. Данный интерфейс широко применяется для рендера трехмерной графики, предоставляет расширенные графические возможности обработки и работы с *3D*-картами текстур и различных эффектов.

Интерфейс *Direct3D* интегрирован с другими технологиями *DirectX,* что позволяет ему обеспечивать аппаратный рендер в двумерных плоскостях наложения, спрайтовую анимацию, отображать видео, предоставлять возможность использования *2D и 3D* графики в интерактивных медиа-связях [5].

Главным конкурентом *Direct3D* считается технология *Open Graphics Library (OpenGL)*.

*OpenGL –* спецификация (документ), определяющий некоторый набор различных функций, а также их поведение.

*Cross-platform Audio Creation Tool* (*XACT3*) *–* библиотека и, по совместительству, движок, предназначенный для программирования аудио, входящий в состав компонентов *DirectX*. С помощью данного интерфейса разработчик может создавать и воспроизводить аудио информацию. Данная библиотека пришла на смену библиотеке *DirectSound,* которая устарела с выходом *DirectX 8.*

*DirectX Graphics Infrastructure* (*DXGI*) – это интерфейс, который отвечает за сопоставление между интерфейсами *DirectX* и графическим ядром*.* Графическое ядро, в свою очередь, взаимодействует с драйвером дисплея *Windows*. В возможности *DXGI* входит:

- перечисление графических адаптеров и мониторов;

- перечисление режимов отображения;

- выбор форматов буферов;

- совместное использование ресурсов между процессорами.

*DirectCompute –* это интерфейс, основная задача которого – выполнение различного рода вычислений с использованием графических процессоров.

*DirectX Raytracing* (*DXR*) *–* это функция, которая реализует трассировку лучей, для видео рендеринга. *DXR* начинается с отправки луча из каждого пикселя в заданной плоскости и вычисляет, какие объекты на плоскости попадают под луч в первую очередь. Затем алгоритм *DXR* оценивает количество света в том месте, где луч пересекает объект, и прикрепляет это вычисление к объекту. Объекты могут иметь разные свойства, которые будут поглощать или отражать свет с разной скоростью. Чтобы предотвратить бесконечные отскоки луча, *DXR* остановит вычисления после определенного количества пересечений. Лучи, которые никогда не взаимодействуют с объектом, отслеживаются по тому, как далеко они ушли. Это делается для того, чтобы алгоритм понимал, когда лучи зашли слишком далеко [6].

В результате применения данной технологии, пользователь получает качественное изображение (рисунок 1.1).



Рисунок 1.1 – Пример сцены, отрисованной с помощью технологии трассировки лучей

*DirectStorage –* интерфейс для работы с быстрыми *SSD.*

### **1.1.3** Интерфейс Direct2D

Для создания игры «Бег с препятствиями» наиболее оптимальным решением будет использовать именно комопнент *2D* графики. Верным выбором в таком случает будет служить интерфейс *Direct2D.*

*Direct2D* является аппаратным ускорением *2D* графики, который обеспечивает высокую производительность и высокое качество визуализации для *2D* геометрии, растровых изображений и текста. *Application Programming Interface* (*API*) *Direct2D* предназначен для качественного взаимодействия с *GDI*, *GDI+* и *Direct3D* [7].

Интерфейс *Direct2D* позволяет пользователям визуально представить ресурсы, которые они могут создать в любом векторном редакторе графики, без предварительной конвертации данных ресурсов в изображения растрового формата. Такая уникальная возможность обеспечивается благодаря тому, что *Direct2D* способен поддерживать синтаксический анализа и рисование *SVG*-изображения (рисунок 1.2).

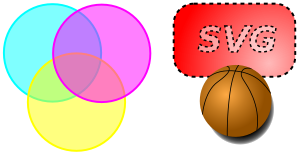


Рисунок 1.2 – Пример *SVG*-изображения

*Scalable Vector Graphics* (*SVG*) – язык разметки векторной графики, который предназначен для описания смешанной векторно-растровой графики в формате *eXtensible Markup Language* (*XML*).

Примечательно, что *Direct2D* поддерживает только ограниченное подмножество *SVG*, которые подходят для изображений, но не поддерживает все возможности рисования *SVG*.

*Direct2D* рендеринг поддерживает более широкий спектр форматов цвета. Данная возможность позволяет пользователям использовать больше типов шрифтов в своих приложениях на базе *Direct2D.*

Начиная с *Windows10*, *Direct2D* обеспечивает поддержку создания и рендеринга пакетов спрайтов. Спрайт – графический объект, применяемый в компьютерной графике. Чаще всего, спрайт (рисунок 1.3) представляет собой растровое изображение, которое можно отобразить на экране.

По сравнению с универсальным методом *DrawImage* (метод библиотеки *System.Drawing*, который рисует указанный пользователем объект *Image* в заданном месте, используя исходный размер изображения), пакеты спрайтов требуют значительно меньших затрат процессора на каждое изображение. Это делает их идеальными для сценариев, включающих сотни или тысячи схожих изображений, таких как игровые спрайты или системы частиц.

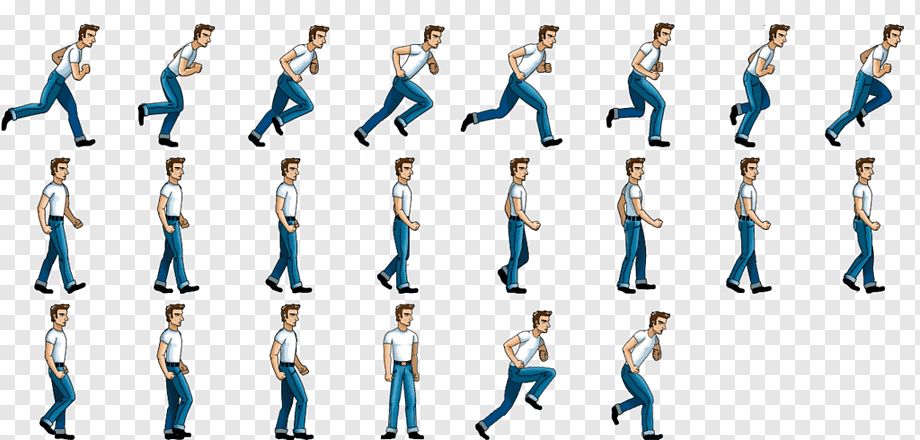


Рисунок 1.3 – Пример *2D* спрайта

*Direct2D* предоставляет новый примитив для представления чернильных штрихов. Штрихи чернил *Direct2D* определяются кривыми Безье, поддерживают различные формы наконечника и преобразования и могут иметь фиксированную или переменную толщину. Встроенная поддержка штрихов чернил *Direct2D* позволяет приложениям легко визуализировать более быстрые и красивые чернила, чем предыдущие подходы, которые обычно требовали, чтобы приложения сами управляли чернилами в виде серии эллипсов и четырехугольников.

Интерфейс *Direct2D* включает в себя богатый набор новых встроенных эффектов, которые отвечают главным запросам пользователей и позволяют создавать новые виды визуальных сценариев. Среди новых эффектов, реализованных с появлением *Windows10,* можно выделить:

1. контрастный эффект (увеличивает или уменьшает контрастность изображения);
2. эффект бликов и теней (настройка бликов и теней изображения);
3. эффект резкости (повышает резкость изображения);
4. эффект оттенков серого (преобразует изображение в монохроматический серый цвет);
5. эффект выпрямления (поворачивает и при необходимости масштабирует изображение);
6. эффект обнаружения краев (отфильтровывает содержимое изображения, оставляя линии по краям контрастных участков изображения);
7. эффект пастеризации (эффект уменьшает количество уникальных цветов в изображении).

## 1.2 Отличительные особенности технологии *DirectX* в сравнении с

## *OpenGL*

Технологии *DirectX* и *OpenGL* являются конкурирующими интерфейсами. Они оба могут использоваться для визуализации *3D* и *2D* компьютерной графики. Отсюда можно выделить одно из основных преимуществ *DirectX:* наличие множества различных компонентов, которые позволяют пользователю работать не только с графикой, но и со звуком, текстом, вычислениями с использованием графических процессоров и памятью компьютера.

Представленные интерфейсы прикладного программирования имеют множество различий. Одним из них можно считать поддержку расширений. Так как *DirectX* является проприетарным продуктом, то и изменения в него вносятся исключительно самой компанией разработчиком – *Microsoft*. *OpenGL*, в свою очередь, является свободным *API,* распространяющийся под лицензией. *OpenGL* представляет собой открытый стандарт, однако некоторые его функции запатентованы. Правки в интерфейс и расширение *OpenGL* осуществляются всем сообществом, в которое входят и ключевые производители видеокарт: *AMD* и *NVIDIA*. В конечном итоге наиболее важные расширения становятся частью основной спецификации. Каждая новая версия *OpenGL* представляет из себя старую версию с добавлением некоторых новых расширений. В то же время новые функции по-прежнему доступны в виде расширений.

Еще одно существенное различие двух технологий – кроссплатформенность. Проприетарный *DirectX* реализован официально лишь в семействе операционных систем *Windows*, включая те версии, которые используются в семействе игровых консолей *Xbox.*

*OpenGL* имеет реализации, доступные на многих платформах , включая *Microsoft* *Windows*, *[Unix](https://en.wikipedia.org/wiki/Unix" \o "Unix)*-системы, такие как *Mac*[*OS*](https://en.wikipedia.org/wiki/OS_X), *[Linux](https://en.wikipedia.org/wiki/Linux" \o "Линукс)*. *Nintendo* и *Sony* разработали свои собственные библиотеки, которые похожи, но не идентичны *OpenGL*. *OpenGL* было выбрано в качестве основной графической библиотеки для *[Android](https://en.wikipedia.org/wiki/Android_(operating_system)" \o "Android (операционная система))*, *[BlackBerry](https://en.wikipedia.org/wiki/BlackBerry" \o "ежевика)*, *[iOS](https://en.wikipedia.org/wiki/IOS" \o "IOS)* и *[Symbian](https://en.wikipedia.org/wiki/Symbian" \o "Symbian)* в форме *[OpenGL ES](https://en.wikipedia.org/wiki/OpenGL_ES" \o "OpenGL ES)*.

*DirectX* основан на технологии *COM,* одним и преимуществ которой является то, что *API* можно использовать на любом *COM*-совместимом языке, в частности: *[Object Pascal](https://en.wikipedia.org/wiki/Object_Pascal" \o "Объект Паскаль)* (*[Delphi](https://en.wikipedia.org/wiki/Delphi_(programming_language)" \o "Delphi (язык программирования))*), *С++*, [*C#*](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) и др.

*OpenGL* – спецификация, реализованная на [языке программирования](https://en.wikipedia.org/wiki/Programming_language) [*C*](https://en.wikipedia.org/wiki/C_(programming_language)), которая может быть использована и на других языках. Как *API*, *OpenGL* не зависит ни от одной функции языка программирования и может быть вызван практически из любого языка. В *OpenGL* используется так называемая машина состояний (конечный автомат). Результат вызовов функций *OpenGL* зависит от внутреннего состояния, и может изменять его. В *OpenGL*, чтобы получить доступ к конкретному объекту (например, текстуре), нужно сначала выбрать его в качестве текущего функцией *glBindTexture*, а затем уже можно влиять на объект, например, задание содержимого текстуры осуществляется вызовом *glTexImage2D*.

На рынке профессиональной графики более востребованным считался *OpenGL,* нежели *DirectX,* в то время как набор интерфейсов прикладного программирования от *Microsoft* считался наиболее пригодным в основном для компьютерных игр. Однако, на данный момент как *OpenGL*, так и *DirectX* имеют достаточно большое перекрытие в функциональности, которое может быть использовано для большинства общих целей, причем операционная система часто является главным критерием, диктующим, что используется: *DirectX* является общим выбором для *Windows*, а *OpenGL* – почти для всего остального.

При рассмотрении игровой индустрии можно с уверенностью убедиться, что большинство производителей игровых приложений отдают свое предпочтение технологии *DirectX* благодаря гибкости и расширенному набору библиотек. Игр, где присутствует полноценная поддержка *OpenGL* не так уж и много. Производители игровых приставок оснащают свои продукты собственными *API* для максимизации производительности, что делает сравнение *OpenGL* и *DirectX* актуальным лишь для платформы ПК.

Таким образом менее продвинутый и реже обновляемый *DirectX* является наиболее предпочтительным для тех, кто использует операционную систему *Windows* и стремится получить максимальную производительностьво время игрового процесса.

## 1.3 Основы оболочки *SharpDX*

*SharpDX* – это низкоуровневая оболочка для *API* *DirectX*. Большинство ресурсов, доступных в Интернете, в основном предназначены для *C++*, а не для *C#.* Данная оболочка позволяет решить проблему использования языка *C#* и технологии *DirectX*.

Перед началом работы, пользователю необходимо обратить внимание на то, что если оболочка *SharpDX* является низкоуровневой для него, то для пользователя рекомендуется рассмотреть возможность использования более высокоуровневого движка или *API*, такого как *Xenko*, *MonoGame* или *Win2D*.

При программировании с помощью технологии *API* *DirectX* разработчику необходимо использовать *COM*-объекты. Данная модель программирования используется несколькими технологиями, включая большую часть *API* *DirectX*.

Все *SharpDX* *COM*-объекты наследуют от *COM*-объекта, который реализует интерфейс *IDisposable* шаблон. Метод *Dispose()* на самом деле является сокращением *COM*-метода *ReleaseReference()*.

Приведем некоторые пространства имен, которые необходимо подключить для решения поставленной задачи.

Пространство имен[*SharpDX*](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.ru&sl=en&tl=ru&u=http://sharpdx.org/documentation/api/n-sharpdx&usg=ALkJrhjqlCt_OwbygDPuFepooPpZxV3c7w) содержит основные классы, используемые *SharpDX*.

Пространство имен [*SharpDX.Design*](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.ru&sl=en&tl=ru&u=http://sharpdx.org/documentation/api/n-sharpdx-design&usg=ALkJrhixTC54YrWKM4-TjJgI1MqjgW_2Iw) содержит преобразователи времени разработки для всех векторов структур.

Пространство имен [*SharpDX.Direct2D1*](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.ru&sl=en&tl=ru&u=http://sharpdx.org/documentation/api/n-sharpdx-direct2d1&usg=ALkJrhjJwMvMlVZ-OOiYYy9SMWBqS4aRnw) предоставляет управляемую *Direct2D* *API*.

*SharpDX*.*Windows* содержит *System.Windows.Forms* содержит основные элементы управления, используемые для отображения контента *DirectX*.

## 1.4 Пример игрового приложения на тему «Бег с препятствиями»

В качестве примера игрового приложения, реализующего 2D игру на тему «Бег с препятствиями» прекрасно подойдет, знакомая каждому пользователю сети интернет, игра «Хром Динозавр» (рисунок 1.4).

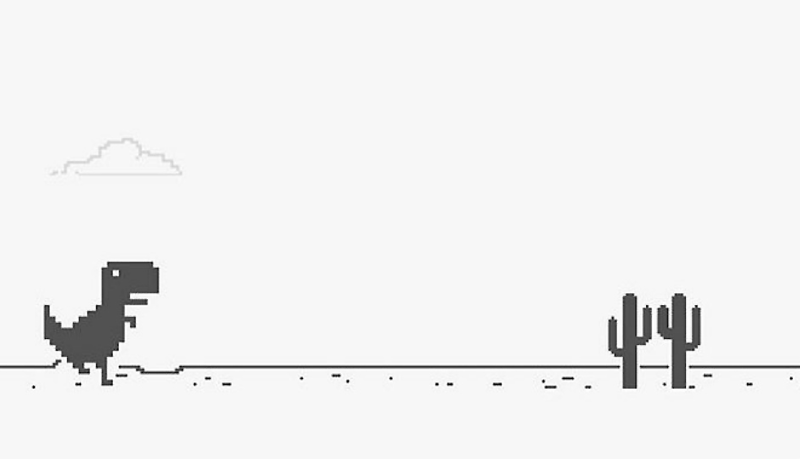


Рисунок 1.4 – Игровое пространство игры «Гугл Динозавр»

Идея создания «бесконечно бегущего динозавра» для развлечения пользователь, оказавшихся без доступа в сеть интернет появилась в начале 2014 года.

Примечательно, что именно визуальное оформление игры (кактусы и пустыня) отсылают пользователя к самым первым вариациям страницы «без доступа к сети». Именно такое оформление было характерно для страницы тех времен.

Разработчики утверждают, что каждый месяц игра запускается около 270 миллионов раз, как на компьютерах, так и на мобильных устройствах. Создатели обращают внимание на то, что большая часть запусков игры приходится на страны, где доступ в сеть интернет является либо слишком дорогим, либо ненадежным. К таким странам относятся Индию, Мексику и Индонезию.

Игра «Хром Динозавр» не имеет определенной конечной цели. Пользователю предлагается скоромный набор управляющий клавиш для того, чтобы поставить максимально возможный рекорд. На пути «динозавра» встречаются различные препятствия, и опасности в виде внутриигрового ландшафта и фауны (кактусы, летающие птицы), что делает данный продукт идеальным примером игры на тему «Бег с препятствиями».

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Попов, А. А. *DirectX* – это просто. Программируем графику на *С++*/ А.А. Попов – СПб.:БХВ-Петербург, 2008. – 7 c.
2. Игнатенко, А. *OpenGL* и *DirectX*: взгляд изнутри/ А. Игнатенко // Компьютерная графика и мультимедиа. – 2004. – №2. – С.1 – 2.
3. *Habr* [Электронный ресурс]. — 2021. — Режим доступа: https://habr.com/ru/post/499152/ — Дата доступа: 25.02.2021.
4. *Microsoft Docs* [Электронный ресурс]. — 2021. — Режим доступа: https://docs.microsoft.com/en-us/windows/win32/prog-dx-with-com — Дата доступа: 26.02.2021.
5. *Direct3D* [Электронный ресурс]. — 2021. — Режим доступа: <https://ru.qaz.wiki/wiki/Direct3D> — Дата доступа: 25.02.2021.
6. *DirectX Raytracing* [Электронный ресурс]. — 2021. — Режим доступа: <https://ru.qaz.wiki/wiki/DirectX_Raytracing> — Дата доступа: 25.02.2021.
7. *Microsoft Docs* [Электронный ресурс]. — 2021. — Режим доступа: <https://docs.microsoft.com/en-gb/windows/win32/direct2d/> — Дата доступа: 02.03.2021.