МИНИСТЕРСТВО ОБРАЗОВАНИЯ КРАСНОЯРСКОГО КРАЯ
КРАЕВОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«АЧИНСКИЙ ТЕХНИКУМ НЕФТИ И ГАЗА ИМ Е.А. ДЕМЬЯНЕНКО»

Специальность 09.02.07 Информационные системы и программирование

Группа ИСП-20/9(2)

КУРСОВАЯ РАБОТА

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем»

Тема: Разработка компьютерной игры «Arcade: Duck Hunt»

Выполнил: _____ В.М. Соломатов

Проверил: _____ Д.А. Павлов

Ачинск, 2022

СОДЕРЖАНИЕ

Оглавление

ВВЕДЕНИЕ

Игра – непременный спутник развития человека. На стадии археокультуры игры выполняли чрезвычайно важные функции. Они использовались для социализации подрастающего поколения, подготовки к коллективной охоте, для тренировки.

Ценность игр заключается в том, что они создаются в обучающих целях. Благодаря их использованию можно добиться более прочных и осознанных знаний, умений и навыков.

Развитие информационных технологий дает возможность создавать компьютерные игры, которые максимально приближенны к реальным ситуациям, тем самым, делая процесс обучения более эффективным. Также из-за развития информационных технологий компьютерные игры может запустить каждый человек, у которого есть компьютер, телефон или игровая приставка.

Целью моей курсовой работы является разработка компьютерной игры «Arcade: DuckHunt».

Аркада (англ. *arcade game, arcade genre*) — жанр компьютерных игр, характеризующийся коротким по времени, но интенсивным игровым процессом.

Аркада характеризуется следующими свойствами:

- Игра на одном экране. Так, игроки в любой момент времени могли видеть весь игровой мир и принимать решения, исходя из полной информации о его состоянии.
- Бесконечная игра. Это влияло на то, что игроки делали вызов сами себе - насколько долго они смогут продержаться.
- Множество жизней. Такой подход позволяет новичкам получить большую возможность изучить игровые механики до того, как игра заканчивается. Если игрок лучше понимал игру, то возрастала вероятность того, что он вернётся к ней снова.
- Игровой счет. Очки позволяют игроку понять, насколько хорошо он играл, несмотря на то, что выиграть невозможно.

Для разработки компьютерной игры «Arcade: DuckHunt» и достижения цели курсовой работы были поставлены следующие задачи:

Задачи:

- Придумать концепцию компьютерной игры;
- Составить список спрайтов и нарисовать их;
- Изучить и реализовать процедурную генерацию;
- Реализовать случайное передвижение объекта в пространстве;

- Вхождение координат курсора в зону нахождения объекта «утка»;
- Реализовать перевод одного типа ресурсов в другой тип ресурсов с изменением коэффициента стоимости полученных ресурсов;
- Разработать программную реализацию компьютерной игры «Arcade: DuckHunt»;
- Подключить и реализовать запись данных в базу SQLite;
- Написать руководство пользователя.

1. Общая часть

1.1 Постановка задачи

1.1.1 Название задачи

В ходе выполнения курсовой работы необходимо реализовать игру «Arcade: Duck Hunt» позволяющую:

- развивать скорость реакции
- приобрести аналитические навыки
- развивать мелкую моторику рук
- развивать умение планировать
- получить удовольствие от процесса игры

1.1.2 Математическое описание задачи

Логика игры реализована в двухмерном пространстве, имеющим координаты $X_н$, $Y_н$.

Жанр аркада представляет собой сцену, расположенную по координатам X, Y, на которой создаются объекты по координатам $X_н$, $Y_н$. Игрок может управлять определенным объектом путем изменения его координат на n единиц. Также игрок может изменять параметры управляемого им объекта, путем изменение определенных констант на k единиц.

1. «Процедурная генерация». Сцена, на которой располагаются объекты, находится в пространстве на координатах $X_н$, $Y_н$. На сцене случайно генерируются объекты с координатами в диапазоне от Y_1 доY_2.
2. «Движение объекта». После генерации, объекты двигаются на n ед., как по оси минус $X_н$, так и по оси $X_н$.
3. «Стрельба». Игрок может уничтожать эти объекты, при условии, что координаты $X_н$, $Y_н$ его курсора совпадут с координатами $X_н$, $Y_н$ зоны, закрепленный за каждым созданным объектом. При уничтожении объекта игроком, переменная score увеличивается на 1, увеличивая рейтинг игрока.
4. «RandomFly». Объекты, расположенные по координатам $X_н$, $Y_н$, могут изменить свое направление по координате $Y_н$, в случайное значение t и в случайное значение $X_н$.
5. «Выбор оружия». Объект «Оружие» можно выбирать, изменяя логические переменные a, b, c по нажатию UI-кнопок на сцене, расположенной по координатам $X_н$, $Y_н$.

6. «Приготовление утки». Значение особых объектов g можно увеличивать на 1, спустя определенное время, заданное в переменной t.

7. «Увеличение сложности». В зависимости значения Δt объекты, увеличивается частота генерации на n ед. и скорость движения S на nед.

8. «Продажа». Переменная countDuck уменьшается на 1 ед., переменная Money увеличивается на n ед., при нажатии на определенные UI-кнопки.

9. «Разделывание уток». Переменная countDuck. Переменная countDuck уменьшается на 1 ед., а переменные pero, grudka, krilo увеличиваются на случайное значение n.

10. Запись значений переменных Score, Money в базу данных SQLite.

11. Вывод значений переменных Score, Money из базы данных SQLite.

12. «Фонарик». Создается окружность, координаты которой соответствуют координатам курсора мыши.

13. «Смена дня и ночи». Значение переменной источника света изменяется на число n ед., затемняя область сцены, и увеличивается на n ед., осветляя область сцены

14. «Смерть». При попадании координат курсора в зону координат Утки, после выстрела, утка меняет свое направление по оси Y, изменяя скорость на u ед., движется вниз.

15. «Увеличение счета в ночной период». В диапазоне значений переменной источника света N_1 и N_2, переменная score умножается на 2.

16. «Скорость». При создании объекта «Утка», ему присваивается случайное значение переменной, отвечающая за скорость объекта.

1.1.3 Требования предъявляемые модели

К математической модели относятся следующие требования:

- Адекватность. Модель считается адекватной, если отражает заданные свойства с приемлемой точностью. Точность определяется как степень совпадения значений выходных параметров модели и объекта. Точность модели различна в разных условиях функционирования объекта.
- Универсальность. Определяется в основном числом и составом учитываемых в модели внешних и выходных параметров.
- Экономичность. Модель характеризуется затратами вычислительных ресурсов для ее реализации — затратами машинного времени и памяти.

- Простота. Модель, при которой желаемый результат достигается за то же время с той же точностью при учете меньшего количества факторов при расчете, называется простой.
- Потенциальность (предсказательность). Возможность получения новых знаний с помощью применения модели.
- Достаточная точность результатов решения задачи, надежность функционирования модели. Способность к совершенствованию модели без ее коренной переделки.
- Простота форм исходных данных и их заполнения при выдаче задания на расчет.

К компьютерной модели относятся следующие требования:

- Реализовать сохранение счета и баланса игрока в базу данных SQLite.
- Сделать меню, где будут расположены кнопки «Играть», «Таблица лидеров», «Выход».
- Реализовать игровые механики.
- Сделать ввод игрового имени пользователя при входе в игру, чтобы записать его счет в базу данных.

1.2 Описание языка программирования

C# был разработан Microsoft в 2000 году, чтобы предоставить современный язык программирования общего назначения, который можно использовать для разработки всех видов программного обеспечения, предназначенного для различных платформ, включая Windows, Web и Mobile, используя только один язык программирования. За последние 10 лет C# не покидает топ-5 самых используемых языков программирования в мире. Миллионы разработчиков программного обеспечения используют C# для создания всех видов программного обеспечения.

C# является основным языком для создания программных приложений Microsoft .NET. Разработчики могут создавать практически все виды программного обеспечения с использованием C#, включая приложения пользовательского интерфейса Windows, консольные приложения, серверные службы, облачные API, веб-службы, элементы управления и библиотеки, серверные приложения, веб-приложения, нативные приложения для iOS и Android, программное обеспечение для искусственного интеллекта и машинного обучения, а также блокчейн приложений.

C# с помощью Visual Studio IDE обеспечивает быструю разработку приложений. C# — это современный, объектно-ориентированный, простой, универсальный и

ориентированный на производительность язык программирования. C# разработан на основе лучших функций и вариантов использования нескольких языков программирования, включая C ++, Java, Pascal и SmallTalk.

Unity - межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

1.3 Обоснование выбора языка программирования

Синтаксис C# похож на C ++. Библиотека .NET и C# похож на Java. C# поддерживает современные возможности объектно-ориентированного языка программирования, включая абстракцию, инкапсуляцию, полиморфизм и наследование. C# является строго типизированным языком, и большинство типов наследуются.

C# поддерживает концепции классов и объектов. Классы имеют такие элементы, как поля, свойства, события и методы. Вот подробная статья о C# и ООП.

C# универсален, современен и поддерживает современные потребности программирования.

В Unity используется среда .NET и язык программирования C# — самый популярный в разработке игр. В отличие от других движков, где используется C++, в C# существует много элементов и приемов, которые уже реализованы, и программисту нужно только воспользоваться ими.

Движок компилирует код C# для каждого целевого устройства, поэтому вы можете развертывать приложения для ПК, мобильных устройств, консолей, а также платформ AR и VR.

Backtrace для Unity позволяет автоматически отслеживать ошибки и сбои на каждой платформе.

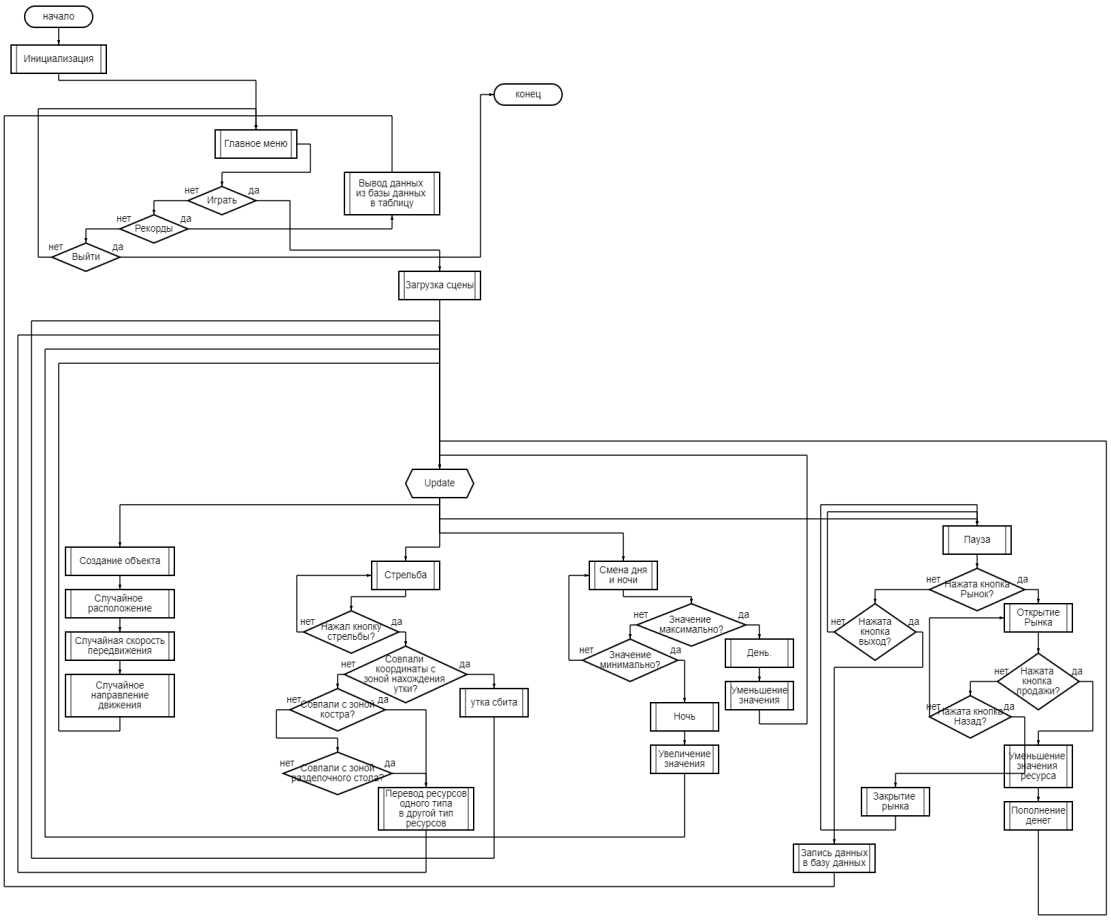
Unity поддерживает быстрое прототипирование и масштабируемые конвейеры ассетов в настраиваемом редакторе.

Редактор Unity используется для создания миров, анимации и игровых сцен, а также рендеринга. Благодаря поддержке таких инструментов, как Maya и Blender.

Unityимеет AssetStore, где имеется огромное количество различных, как бесплатных, так и платных плагинов и ресурсов для создания игр. Все они собраны в одном месте с удобным поиском и возможностью загрузить, интегрировать и получить сразу рабочий функционал.

2. Специальная часть

2.1 Описание алгоритмической модели



2.1.1 Входные данные

Компьютерная мышь — координатное устройство для управления курсором и отдачи всевозможных указаний компьютеру. Руководство курсором осуществляется путём передвижения мыши по поверхности.

Компьютерная клавиатура — устройство, позволяющее пользователю вводить информацию в компьютер (устройство ввода). Представляет собой набор кнопок, размещенных в определённом порядке.

Чтение данных – это передача данных из внешней памяти в оперативную память. приложение, связанное с необходимостью хранения и обработки значительных объемов данных, неизбежно вынуждено хранить эти данные в файлах, а значит и с записью данных в файл и чтением их из файла.

2.1.2 Выходные данные

В данном проекте выходными данными являются:

- звуковое сопровождение - любые мелодии, композиции или саундтреки видеоигр.
- сохранение данных предполагает запись данных в базу данных SQLite для дальнейшего использования.
- графическое представление реализуется через компонент программного интерфейса приложения Direct 12, обеспечивающий функции для взаимодействия операционной системы и приложений с драйверами видеокарты.

2.2 Целевое назначение процедур и функций

В разработанной компьютерной игре «Arcade: Duck Hunt» используются методы, представленные в таблице 1.

Таблица 1 – Методы

Название	Описание
GetDown()	Меняет направление движение утки, изменяя вектор движения по Y, выполняет объявленные в нем методы.
DisappearSec(float time)	Отсчитывает переданное значение и уничтожает передававемый в метод Destroy объект
PlaySound(float time)	Отсчитывает переданное значение и проигрывает переданный в AudioSource звук
ChangeDirection(float time)	Отсчитывает переданное значение и проверяет условие, при верности которого, обновляет значение переменной ySpeed на положительное.
Update() класса MouseFollow	Присваивает переменной mousePos значение позиции курсора на экране.
FixedUpdate() класса MouseFollow	Присваивает объекту класса Vector2 разность переменной mousePos и значение новосозданного объекта класса Vector2. Присваивает переменной angle значение угла умноженное на константу минус 90. Проверяет

	логическую переменную, если true, то присваивает значение угла поворота значению поворота объекта. Если false, то присваивает значение угла поворота в реальном времени, высчитанное через круги Эйлера.
Start() класса MouseFollow	Передает компонент Rigidbody2D объекту класса Rigidbody2D.
Start() класса Database	Создает объект класса SqliteCommand, передает в этот объект путь к базе данных SQLite и возвращающий набор данных из таблицы Liders базы данных. Вызывает метод setConnection()
setConnection()	Передает в переменную путь к базе данных, обновляет переменную dbconnection, осуществляет чтения данных из базы данных.
UpdateConnection()	обновляет таблицу в базе данных на переданные значения переменных из статичного класса, записывая их значения в соответствующие столбцы в таблице.
PlayGame()	Загружает сцену с игровым процессом.
ExitToMenu()	Загружает сцену с Главным меню.
ExitGame()	Закрывает приложение.
Store()	Перемещает панель, на координаты $X_н$, $Y_н$.
BackStore()	Перемещает панель, на начальные координаты $X_н$, $Y_н$.
SellDuck()	Проверяет значение переменной. Если оно не равно нулю, отнимает от этого значения 1, прибавляет к переменной, содержащую количество накопленных монет, 100, записывает в TextMesh строку с выводом значения обоих переменных.
Start() класса GameController	Вызывает метод Resume(). Делает неактивными переданные объекты через свойство SetActive. Обнуляет

	все статичные переменные в классе StaticClass.
Update() класса GameController	Цикл, проверяющий, нажата ли кнопка паузы, проверяет переменную notgameover, проверяет время до создания следующего объекта "Утка", а после выполняет метод SpawnDuck() и Shoot().
Shoot()	Обрабатывает нажатие мыши, проверяет совпадение координат курсора и координаты коллайдера объекта.
SpawnDuck()	Создает объект «Утка» с определенным размером, со случайной позицией по оси Y, присваивает случайное значение переменной скорости утки.
Resume()	Устанавливает свойство SetActive у объекта pauseMenuUI на false, изменяет значение Time. TimeScale на 1, логической переменной isPaused присваивает false.
Pause()	Устанавливает свойство SetActive у объекта pauseMenuUI на true, изменяет значение Time. timeScale на 0, логической переменной isPaused присваивает true.
OnMouseDown()	Изменяет значения логических переменных, уменьшает размер UI-кнопки.
OnMouseUp()	Увеличивает размер UI-кнопки.
Start() класса DayNight	Присваивает значение перменной AlphaA.
Update() класса DayNight	Проверяет значение AlphaA, если ноль, то увеличивает параметры RGB в положительную сторону до макимального значения, если единица, то увеличивает параметры RGB в отрицательную сторону до минимального значения.
OnMouseDown() класса BofireController	проверяет значение статичной переменной CountDuck, если не равна нулю, то увеличивает значение статичной переменной на 1 CookingDuck, уменьшает значение CountDuck на 1.

OnMouseDown() класса Razdelka	Уменьшает значение CountDuck на 1. Присваивает случайное значение от 0 до 3 переменной i. Проверяет значение i и выполнет case в соответсвии со значением i.
SellPero()	Уменьшает количество перьев и прибавляет определенное значение к переменной Money.
SellGrudka()	Уменьшает количество грудок и прибавляет определенное значение к переменной Money.
SellLopast()	Уменьшает количество крыльев и прибавляет определенное значение к переменной Money.
SellCook()	Уменьшает количество приготовленных уток и прибавляет определенное значение к переменной Money.

2.3 Инструкция по выполнению программы

Наименование программы: Arcade: Duck Hunt

Краткое описание программы: Arcade: Duck Hunt – представляет собой аркаду, имеющая игровой процесс, основанный на стрельбы по уткам из ружья, а также на продаже уток на рынке.

Перечень выполняемых программой функций:

- создание среды в обучающих целях
- укрепление навыков, умений и знаний
- методы для решения поставленных задач:
- теоретический анализ
- анализ электронных источников информации
- встраиваемая база данных SQLite, в которую записываются игровые переменные.

2.4 Руководство пользователя

Для запуска игры два раза нажмите на одноимённом .exe файле в папке с игрой.

После запуска программы, на экране появится «Главное меню», откуда вы сможете: начать игру, нажав кнопку «ИГРАТЬ», посмотреть таблицу лидеров, нажав кнопку «РЕЙТИНГ», закрыть программу, нажав кнопку «ВЫХОД». После нажатия кнопки «ИГРАТЬ» на экране отобразится сцена, где игрок с помощью мыши должен наводить курсор на утку и нажимать Левую кнопку мыши (далее ЛКМ), после чего утка будет считаться сбитой. При нажатии ЛКМ по внутриигровым кнопкам выбора оружия,

пользователь может изменить тип используемого оружия. При нажатии на клавиатуре клавиши «Escape» (далее Esc), откроется меню паузы, где расположены кнопки «Рынок» и «Выход». При нажатии кнопки «Рынок», на экране появится окно с кнопками «Продать утку», «Продать перья», «Продать приготовленную утку». Нажав на одну из них, пользователь получит указанное под кнопкой, количество монет, если таковой предмет, указанный в названии кнопки, у него есть. Если такого предмета нет - пользователь ничего не получит.

Запросы от программы отсутствуют.

В исключительных ситуациях и при появлении ошибок, рекомендуется подождать ответа от программы. Если ответ от программы не пришел, нажмите сочетание клавиш Altи F4 или откройте «Диспетчер задач» с помощью сочетания клавиш Ctrl + Alt + Delete, выделите процесс с названием программы и нажмите «Снять задачу». После этих действий запустите приложение снова.

Чтобы закончить работу с программой, следует нажать «Esc» во время игры, после выйти в главное меню, нажав кнопку «Выход». После отображения на экране «Главное меню» нажать кнопку «Выход».

Минимальные системные требования:

- процессор: i5 - 9100;
- оперативная память объемом: 4 гб;
- операционная система: Windows 10;
- свободное место не менее 1 гб;
- видеокарта: GTX 750.

ЗАКЛЮЧЕНИЕ

При анализе существующих разработок, был проведено их сравнение и выделены их достоинства и недостатки. В ходе анализа стало ясно, что при разработке компьютерной игры с простой игровой механикой, стоит обратить внимание на дополнительные элементы игры, такие как сюжет и графическое оформление. Это нужно для того, что бы удержать потенциального игрока и продлить жизненный цикл разработки.

Основываясь на всей полученной в ходе исследования информации, было решено разработать прототип аркады для одного игрока на игровом движке Unity. Такое решение было принято по нескольким причинам:

- по игровой механике, игра жанра платформер проще реализуется;
- игровой движок Unity распространяется бесплатно и позволяет разрабатывать приложения на языке программирования C#.

В ходе выполнения курсовой работы была разработана компьютерная игра «Arcade: Duck Hunt» на языке программирования C# в среде разработки

В программе реализовано выполнение следующих функций: придумана концепция компьютерной игры, составлен список спрайтов и 3D-моделей, нарисованы спрайты с помощью графического редактора Adobe Photoshop, сконструированы 3D-модели в открытом программном обеспечении для создания трехмерной графики Blender.

Также была изучена и реализована процедурная генерация объектов в двухмерном пространстве, реализовано случайное передвижение объекта в двухмерном пространстве, была реализована проверка вхождения координат курсора в зону нахождения объекта, реализован перевод одного типа ресурсов в другой тип ресурсов с изменением коэффициента стоимости полученных ресурсов.

Успешно подключена база данных SQLite и реализована запись в базу данных и вывод данных из нее.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ 2.104–2006 «Единая система конструкторской документации. Основные надписи»;
2. ГОСТ 2.105–95 «Единая система конструкторской документации. Общие требования к текстовым документам»;
3. ГОСТ 2.201–80 «Единая система конструкторской документации. Обозначение изделий и конструкторских документов»;
4. ГОСТ 2.301–68 «Единая система конструкторской документации. Форматы»;
5. ГОСТ 2.302–68 «Единая система конструкторской документации. Масштабы»;
6. ГОСТ 2.316–2008 «Единая система конструкторской документации. Правила нанесения надписей, технических требований и таблиц на графических документах. Общие положения»;
7. ГОСТ 2.321–84 «Единая система конструкторской документации. Обозначения буквенные»;
8. ГОСТ 2.501–2013 «Единая система конструкторской документации. Правила учета и хранения»;
9. ГОСТ 2.701–2008 «Единая система конструкторской документации (ЕСКД). Схемы. Виды и типы. Общие требования к выполнению»;
10. ГОСТ 28388–89 Система обработки информации. Документы на магнитных носителях данных.
11. ГОСТ 3.1102–2011 «Единая система технологической документации. Стадии разработки и виды документов. Общие положения»;
12. ГОСТ 2.102–2013 «Единая система конструкторской документации. Виды и комплектность конструкторских документов»;
13. ГОСТ 19.505-79. ЕСПД. «Руководство оператора».
14. Прайс, М. Дж. C# 7 и .NET Core : кросс-платформерная разработка для профессионалов / Марк Дж. Прайс; [пер. с англ. М. Сагалович, С. Черников] – 3-е изд. – Санкт-Петербург [и др.] : Питер, 2018. – 636 с. : ил. – (Библиотека программиста). УДК 004.438C#:004.42 ББК 3;
15. Unity в действии. Мультиплатформенная разработка на C#. 2-е межд. изд. — СПб.: Питер, 2019. — 352 с.: ил. — (Серия «Для профессионалов»). ISBN 978-5-4461-0816-9 ББК 32.973.2-018 УДК 004.42

16. Unity и C#. Геймдев от идеи до реализации. 2-е изд. — СПб.: Питер, 2019. — 928 с.: ил. — (Серия «Для профессионалов»). ISBN 978-5-4461-0715-5 ББК 32.973.23-018.9 УДК 004.388.4

17. URL: https://github.com/t4guw/100-Unity-Mechanics-for-Programmers/tree/master/programs/raycast_shooting_2d (Дата обращения: 03.11.2022)

18. URL: https://ru.wikipedia.org/wiki/Аркада_(жанр) (Дата обращения: 17.10.2022)

19. URL: http://esate.ru/uroki/OpenGL/uroki-OpenGL-c-sharp/sintaksis-c-sharp/ (Дата обращения: 02.11.2022)

20. URL: https://unity.com/ru (Дата обращения: 04.10.2022)

21. URL: https://unity.com/ru/learn (Дата обращения: 04.10.2022)

22. URL: https://habr.com/ru/company/piter/blog/593237/ (Дата обращения: 03.11.2022)

23. URL: https://github.com/t4guw/100-Unity-Mechanics-for-Programmers/tree/master/programs/main_menu(Дата обращения: 03.11.2022)

24. URL: https://habr.com/ru/post/149356/ (Дата обращения: 09.12.2022)

25. URL: https://www.sqlite.org/index.html (Дата обращения: 05.10.2022)

26. URL: https://github.com/t4guw/100-Unity-Mechanics-for-Programmers/tree/master/programs/pause_game (Дата обращения: 17.11.2022)

27. URL: https://habr.com/ru/post/442954/ (Дата обращения: 17.11.2022)

28. URL: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html (Дата обращения: 22.11.2022)

29. URL: https://github.com/t4guw/100-Unity-Mechanics-for-Programmers/tree/master/programs/load_next_scene (Дата обращения: 30.11.2022)

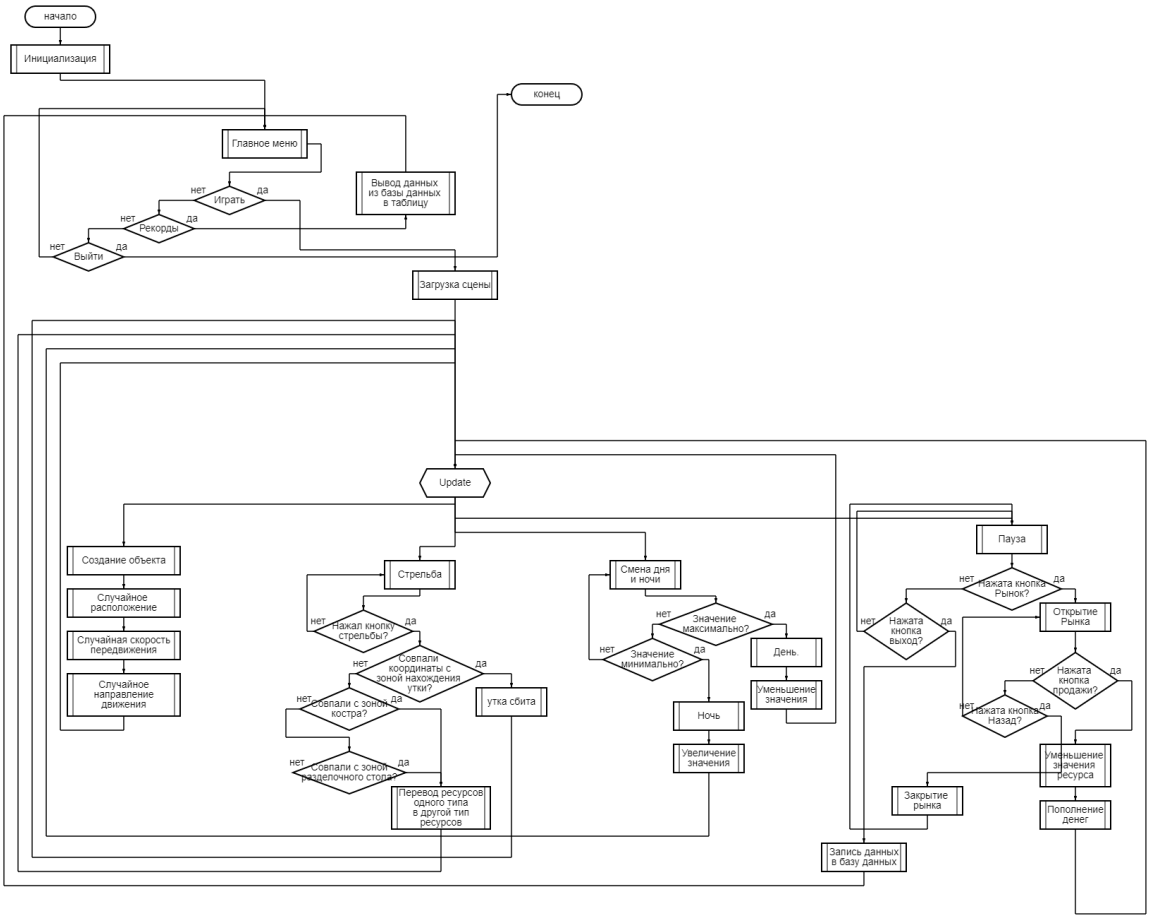
30. URL: https://github.com/t4guw/100-Unity-Mechanics-for-Programmers/tree/master/programs/switch_weapons (Дата обращения: 02.10.2022)

31. URL: https://habr.com/ru/post/460259/ (Дата обращения: 29.11.2022)

32. URL: https://docs.unity3d.com/Manual/UIToolkits.html (Дата обращения: 01.12.2022)

ПРИЛОЖЕНИЕ А

СХЕМА АЛГОРИТМА (БЛОК-СХЕМА)



ПРИЛОЖЕНИЕ В

ТЕКСТ ПРОГРАММЫ (С КОММЕНТАРИЯМИ)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ButtonAR : MonoBehaviour
{
    public GameObject shotgun;
    public GameObject argun;
    public GameObject rpg;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
    public void OnMouseDown()
    {
        transform.localScale = new Vector3(0.9f, 0.9f, 0.9f);
        shotgun.SetActive(false);
        argun.SetActive(true);
        rpg.SetActive(false);
    }
    public void OnMouseUp()
    {
        transform.localScale = Vector3.one;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ButtonRPG : MonoBehaviour
{
    public GameObject shotgun;
    public GameObject argun;
    public GameObject rpg;
    public void OnMouseDown()
    {
        transform.localScale = new Vector3(0.9f, 0.9f, 0.9f);
        shotgun.SetActive(false);
        argun.SetActive(false);
        rpg.SetActive(true);
    }
    public void OnMouseUp()
    {
        transform.localScale = Vector3.one;
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ButtonShotgun : MonoBehaviour
{
    public GameObject shotgun;
    public GameObject argun;
    public GameObject rpg;
    // Start is called before the first frame update

    public void OnMouseDown()
    {
        transform.localScale = new Vector3(0.9f, 0.9f, 0.9f);
        shotgun.SetActive(true);
        argun.SetActive(false);
        rpg.SetActive(false);
    }
    public void OnMouseUp()
    {
        transform.localScale = Vector3.one;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DuckController : MonoBehaviour
{
    public float xSpeed = 0;
    public float xSpeedrandom = 0.25f;
    private float ySpeed = 0;
    private bool isAlive = true;
    private SpriteRenderer theSpriteRenderer;
    Collider2D m_Collider;
    void Start()
    {
        theSpriteRenderer = GetComponent<SpriteRenderer>();
        m_Collider = GetComponent<Collider2D>();
        xSpeed = xSpeed + Random.Range(-xSpeedrandom, xSpeedrandom);
        if (xSpeed >= 0) theSpriteRenderer.flipX = false;
        else theSpriteRenderer.flipX = true;
        StartCoroutine(ChangeDirection(Random.Range(2.5f, 4f)));
        Invoke("Disappear", 20);
    }

    private void LateUpdate()
    {
        transform.Translate(xSpeed * Time.deltaTime, ySpeed * Time.deltaTime, 0f);
    }
    public void SetDirection(float dir)
    {
        xSpeed *= dir;
    }
    public void GetDown() // меняет направление движение утки, изменяя вектор движения по
Y, выполняет объявленные в нем методы.
    {
        isAlive = false;
        m_Collider.enabled = false;
        theSpriteRenderer.flipY = true;
        ySpeed = -1f;
        StartCoroutine(DisappearSec(0.7f));
        Destroy(GetComponent<Animator>());
```

```
        StartCoroutine(PlaySound(0.3f));
    }
    IEnumerator DisappearSec(float time) // отсчитывает переданное значение и уничтожает
передававемый в метод Destroy объект
    {
        yield return new WaitForSeconds(time);
        Destroy(gameObject);
    }
    IEnumerator PlaySound(float time) // отсчитывает переданное значение и проигрывает
переданный в AudioSource звук
    {
        yield return new WaitForSeconds(time);
        GetComponent<AudioSource>().Play();
    }
    IEnumerator ChangeDirection(float time) // отсчитывает переданное значение и
проверяет условие, при верности которого, обновляет значение переменной ySpeed на
положительное.
    {
        yield return new WaitForSeconds(time);
        if (isAlive) ySpeed = Mathf.Abs(xSpeed);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;

public class GameController : MonoBehaviour
{
    [SerializeField] private TextMesh scoreLabel;
    [SerializeField] private TextMesh moneyAndDuck;
    public float spawnSpeed = 3;

    public Light light;
    public Light Sun;

    public GameObject argun;//табло с оружием
    public GameObject rpggun;

    public GameObject shotgun;
    public GameObject ar;//оружие
    public GameObject rpg;

    public GameObject Duck;
    public GameObject pauseMenuUI;

    public float duckDelay;
    public static bool isPaused = false;
    private bool notgameover = true;
    private bool mouseLeftDown = false;
    private Vector2 clickedPos;
    private float nextDuckTime = 0f;
    float pos = 9.2f;
    float dir = 2f;


    void Update()// цикл, проверяющий, нажата ли кнопка паузы, проверяет переменную
notgameover, проверяет время до создания следующего объекта "Утка", а после выполняет
метод SpawnDuck() и Shoot()
    {
        if (Input.GetKeyDown("f"))
        {
```

```
            light.transform.position =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
        }
        if (Sun.color.r <= 50)
        {
            light.enabled = true;
        }
        else
        {
            light.enabled = false;
        }
        if (Input.GetKey("escape"))
        {
            if (isPaused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }

        if (notgameover)
        {
            if (StaticClass.Score == 5)
            {
                argun.SetActive(true);
            }
            else if (StaticClass.Score == 50)
            {
                rpggun.SetActive(true);
            }

            if (Time.time >= nextDuckTime)
            {
                SpawnDuck();
                nextDuckTime = Time.time + duckDelay;
                if (duckDelay >= 1.2f) duckDelay *= .9f;
                if (duckDelay < 1.2f && duckDelay > 0.5f) duckDelay *= 0.99f;

            }
            Shoot();
        }
        else Debug.Log("GAME OVER!");
    }
    public void Shoot() //обрабатывает нажатие мыши, проверяет совпадение координат
курсора и координаты коллайдеры объекта.
    {
        mouseLeftDown = Input.GetMouseButtonDown(0);

        if (mouseLeftDown)
        {
            GetComponent<AudioSource>().Play();
            clickedPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
            RaycastHit2D hit = Physics2D.Raycast(clickedPos, clickedPos, 0f);

            if (hit && hit.collider.gameObject.layer == LayerMask.NameToLayer("Duck"))
            {
                hit.collider.SendMessage("GetDown",
SendMessageOptions.DontRequireReceiver);
                StaticClass.Score++;
                /*
                if (Sun.color.r < 30 && Sun.color.r > 0)
```

```
                {
                    StaticClass.Score++;
                }
                */
                StaticClass.CountDuck++;
                scoreLabel.text = $"Score: {StaticClass.Score}";
                moneyAndDuck.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
            }
        }
    }

    private void Start() // выполняет метод Resume деактивирует GamObject
    {
        Resume();

        StaticClass.Money = 0;
        StaticClass.CountDuck = 0;
        StaticClass.Score = 0;
        StaticClass.Pero = 0;
        StaticClass.CookingDuck = 0;
        StaticClass.Lopast = 0;
        StaticClass.Grudka = 0;

        shotgun.SetActive(true);
        ar.SetActive(false);
        rpg.SetActive(false);
        argun.SetActive(false);
        rpggun.SetActive(false);
    }
    public void SpawnDuck() //устанавливает значение Scale, генерирует случайные
координаты и создает на этих координатах объект "Утка", задавая размер, скорость.
    {
        float newScale = 2f;
        Vector3 randomPositionRight = new Vector3(pos, Random.Range(-0.3f, 2f));
        GameObject newDuck = Instantiate(Duck, randomPositionRight, Quaternion.identity)
as GameObject;
        newDuck.transform.localScale = new Vector3(newScale, newScale, -2f);
        newDuck.SendMessage("SetDirection", dir, SendMessageOptions.DontRequireReceiver);
        dir *= -1;
        pos *= -1;
        Destroy(newDuck, 7.0f);
    }
    public void Resume() // устанвливает pauseMenuUI на false, меняет Time.timeScale = 1f
и устанавливает значение isPaused = false
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        isPaused = false;
    }

    private void Pause() //
    {
        pauseMenuUI.SetActive(true); //устанвливает pauseMenuUI на true, меняет
Time.timeScale = 0f и устанавливает значение isPaused = true
        Time.timeScale = 0f;
        isPaused = true;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
```

```
using UnityEngine;
using Mono.Data.Sqlite;
using Unity.VisualScripting;
using System;

public class MainMenu : MonoBehaviour
{
    public GameObject panelStore;
    [SerializeField] private TextMesh scoreLabel;
    [SerializeField] private TextMesh tableLabel;
    private string path;
    private SqliteConnection dbconnection;
    GameController gc = new GameController();
    public void Start()
    {

    }
    public void PlayGame()//Загружает сцену с игровым процессом.
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    public void ExitToMenu()//Загружает сцену с Главным меню.
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);
    }
    public void ExitGame()//Закрывает приложение.
    {
        Application.Quit();
    }
    public async void TableLiders()// выводит последние сохраненные в базу данных данные
    {
        path = Application.dataPath + "/StreamingAssets/duck.bytes";
        dbconnection = new SqliteConnection("Data Source=" + path);
        await dbconnection.OpenAsync();

        SqliteCommand command = new SqliteCommand("SELECT * FROM [Liders]",
dbconnection);
        using (SqliteDataReader reader = command.ExecuteReader())
        {
            if (reader.HasRows) // если есть данные
            {
                while (reader.Read())   // построчно считываем данные
                {
                    var id = reader.GetValue(0);
                    var username = reader.GetValue(1);
                    var money = reader.GetValue(2);
                    var score = reader.GetValue(3);

                    tableLabel.text = tableLabel.text + $"{id} \t {username} \t Money:
{money} \t Score: {score}\t\n";

                }
            }
        }
    }
    public void Store()//Перемещает панель, на координаты Хн, Yн.
    {
        panelStore.transform.localPosition = new Vector3(0, -1, 0);
    }
    public void BackStore()//Перемещает панель, на начальные координаты Хн, Yн.
    {
        panelStore.transform.localPosition = new Vector3(0, 1080, 0);
    }
```

```
    public void SellDuck()//Проверяет значение переменной. Если оно не равно нулю,
отнимает от этого значения 1, прибавляет к переменной, содержащую количество накопленных
монет, 100, записывает в TextMesh строку с выводом значения обоих переменных.
    {
        if(StaticClass.CountDuck != 0)
        {
            StaticClass.CountDuck--;
            StaticClass.Money = StaticClass.Money + 100;
            scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
        }
        else { scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}"; }

    }
    public void SellPero()//уменьшает количество перьев и прибавляет определенное
значение к переменной Money
    {
        if (StaticClass.Pero != 0)
        {
            StaticClass.Pero--;
            StaticClass.Money = StaticClass.Money + 250;
            scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
        }
        else { scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}"; }
    }
    public void SellGrudka()//уменьшает количество грудок и прибавляет определенное
значение к переменной Money
    {
        if (StaticClass.Grudka != 0)
        {
            StaticClass.Grudka--;
            StaticClass.Money = StaticClass.Money + 350;
            scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
        }
        else { scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}"; }
    }
    public void SellLopast()//уменьшает количество крыльев и прибавляет определенное
значение к переменной Money
    {
        if (StaticClass.Lopast != 0)
        {
            StaticClass.Lopast--;
            StaticClass.Money = StaticClass.Money + 200;
            scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
        }
        else { scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}"; }
    }
    public void SellCook()//уменьшает количество приготовленных уток и прибавляет
определенное значение к переменной Money
```

```
    {
        if (StaticClass.CookingDuck != 0)
        {
            StaticClass.CookingDuck--;
            StaticClass.Money = StaticClass.Money + 150;
            scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
        }
        else { scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}"; }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MouseFollow : MonoBehaviour
{
    public Camera cam;
    Vector2 mousePos;
    public Rigidbody2D rb2d;
    public float rotateSpeed = 50f;
    public bool instant = false;

    private void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        mousePos = cam.ScreenToWorldPoint(Input.mousePosition);
    }

    void FixedUpdate()
    {
        Vector2 lookDirection = mousePos - new Vector2(transform.position.x,
transform.position.y);
        float angle = Mathf.Atan2(lookDirection.y, lookDirection.x) * Mathf.Rad2Deg -
90f;

        if (instant)
        {
            rb2d.rotation = angle;
        }
        else
        {
            Quaternion qTo = Quaternion.Euler(new Vector3(0, -angle, 0));
            transform.rotation = Quaternion.RotateTowards(transform.rotation, qTo,
rotateSpeed * Time.deltaTime);
        }
    }
}




using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class BofireController : MonoBehaviour
{
    [SerializeField] private TextMesh scoreLabel;
    public void OnMouseDown()
    {
        if(StaticClass.CountDuck != 0)
        {
            StaticClass.CookingDuck++;
            StaticClass.CountDuck--;
            scoreLabel.text = $"Money: {StaticClass.Money}, Cook:
{StaticClass.CookingDuck}, Feathers: {StaticClass.Pero}, Chest:
{StaticClass.Grudka}\nDuck's: {StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
        }

    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DayNight : MonoBehaviour
{
    public Light Sun;
    //public GameObject gameObject;
    public int AlphaA;
    void Start()
    {
        AlphaA = 1;
    }

    void Update()
    {
        if(AlphaA == 0)
        {
            Sun.color = new Color(Sun.color.r + 0.01f * Time.deltaTime, Sun.color.g +
0.01f * Time.deltaTime, Sun.color.b + 0.01f * Time.deltaTime, Sun.color.a);
            if (Sun.color.r >= 1.0f)
            {
                AlphaA = 1;
            }
        }
        else
        {
            Sun.color = new Color(Sun.color.r - 0.01f * Time.deltaTime, Sun.color.g -
0.01f * Time.deltaTime, Sun.color.b - 0.01f * Time.deltaTime, Sun.color.a);
            if (Sun.color.r <= 0.0f)
            {
                AlphaA = 0;
            }
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Razdelka : MonoBehaviour
{
    int i = 0;
    [SerializeField] private TextMesh scoreLabel;
    public void OnMouseDown()
```

```
    {
        StaticClass.CountDuck--;
        i = Random.Range(0, 3);
        switch (i)
        {
            case 0: StaticClass.Pero++;
                break;
            case 1: StaticClass.Grudka++;
                break;
            case 2: StaticClass.Lopast++;
                break;
        }
        scoreLabel.text = $"Money: {StaticClass.Money}, Cook: {StaticClass.CookingDuck},
Feathers: {StaticClass.Pero}, Chest: {StaticClass.Grudka}\nDuck's:
{StaticClass.CountDuck}, Wings: {StaticClass.Lopast}";
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StaticClass : MonoBehaviour
{
    public static int Money, Score, CountDuck, Pero, Grudka, Lopast, CookingDuck;
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class Database : MonoBehaviour
{
    private string path;
    private SqliteConnection dbconnection;


    // Start is called before the first frame update
    void Start() // Создает объект класса SqliteCommand, передает в этот объект путь к
базе данных SQLite и возвращающий набор данных из таблицы Liders базы данных.
    {
        SqliteCommand command = new SqliteCommand("SELECT * FROM [Liders]",
dbconnection);
        setConnection();
    }
    public async void setConnection()//Передает в переменную путь к базе данных,
обновляет переменную dbconnection, осуществляет чтения данных из базы данных.
    {
        path = Application.dataPath + "/StreamingAssets/duck.bytes";
        dbconnection = new SqliteConnection("Data Source=" + path);
        await dbconnection.OpenAsync();

        SqliteCommand command = new SqliteCommand("SELECT * FROM [Liders]",
dbconnection);
        SqliteDataReader r = command.ExecuteReader();
        r.Close();

    }
```

```
    public async void UpdateConnection()//обновляет таблицу в базе данных на переданные
значения переменных из статичного класса, записывая их значения в соотвествующие столбцыв
таблице.
    {
        int score = StaticClass.Score;
        int money = StaticClass.Money;
        SqliteCommand command = new SqliteCommand("UPDATE [Liders] SET [Money]=@money,
[Score]=@score WHERE [id]=@Id", dbconnection);
        command.Parameters.AddWithValue("id", 1);
        command.Parameters.AddWithValue("money", money.ToString());
        command.Parameters.AddWithValue("score", score.ToString());
        await command.ExecuteNonQueryAsync();
    }
    public void Insert() //добавить запись имени игрока
    {
        path = Application.dataPath + "/StreamingAssets/duck.bytes";
        string sqlExpression = $"INSERT INTO [Liders] (Money, Score) VALUES
('{StaticClass.Money}', {StaticClass.Score})";
        using (var connection = new SqliteConnection("Data Source=" + path))
        {
            connection.Open();

            // добавление
            SqliteCommand command = new SqliteCommand(sqlExpression, connection);
            int number = command.ExecuteNonQuery();

            sqlExpression = $"UPDATE [Liders] SET Money='{StaticClass.Money}' WHERE
Score={StaticClass.Score}";
            command.CommandText = sqlExpression;
            number = command.ExecuteNonQuery();
        }

    }
}
```