МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ОБРАЗОВАНИЯ ГОМЕЛЬСКОГО ОБЛИСПОЛКОМА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ МАШИНОСТРОИТЕЛЬНЫЙ КОЛЛЕДЖ»

|  |  |
| --- | --- |
|  | Специальность 2-40 01 01 «Программное обеспечение информационных технологий» Специализация 2-40 01 01 35 «Программное обеспечение обработки экономической и деловой информации» |
| Группа Пк-42 |

**ОТЧЕТ**

по производственной технологической практики, пройденной на

|  |
| --- |
| ООО «ОпенМайГейм» |

в период с 06.10.2020 по 30.11.2020

ПДП. 0429961.ОТ

Разработал А.С. Лосев

Руководители практики: Инженер- программист

от предприятия (организации) А.А. Невдашенко

от колледжа преподаватель

Т.В. Завбанова

Гомель 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc57555821)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc57555822)

[2 ОПИСАНИЕ ПРОГРАММЫ 6](#_Toc57555823)

[2.1 Общие сведения 6](#_Toc57555824)

[2.2 Структура программы 6](#_Toc57555825)

[2.3 Входные и выходные данные 7](#_Toc57555826)

[2.4 Вызов и загрузка 8](#_Toc57555827)

[3 ОПИСАНИЕ ПРИМЕНЕНИЯ 9](#_Toc57555828)

[3.1 Требования техники безопасности и охраны труда при эксплуатации программы 9](#_Toc57555829)

[3.2 Энерго и ресурсосбережение 11](#_Toc57555830)

[4 РУКОВОДСТВО ПРОГРАММИСТА 12](#_Toc57555831)

[4.1 Назначение программного обеспечения 12](#_Toc57555832)

[4.2 Характеристики программы 12](#_Toc57555833)

[4.3 Установка программы 12](#_Toc57555834)

[4.4 Сообщения программисту 13](#_Toc57555835)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 14](#_Toc57555836)

[5.1 Наименование программы 14](#_Toc57555837)

[5.2 Условия выполнения программы 14](#_Toc57555838)

[5.3 Выполнение программы 14](#_Toc57555839)

[6 ПРОГРАММА И ДЕТОДИКА ИСПЫТАНИЙ 18](#_Toc57555840)

[6.1 Объект и цель испытаний 18](#_Toc57555841)

[6.2 Порядок испытаний 18](#_Toc57555842)

[ЗАКЛЮЧЕНИЕ 21](#_Toc57555843)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc57555844)

[ПРИЛОЖЕНИЕ А Код программы 23](#_Toc57555845)

[ПРИЛОЖЕНИЕ Б Организационная стуктура организации 35](#_Toc57555846)

**ВВЕДЕНИЕ**

Игра – тип осмысленной непродуктивной деятельности, где мотив лежит не в её результате, а в самом процессе. Также термин «игра» используют для обозначения набора предметов или программ, предназначенных для подобной деятельности.

Сегодня наука больше изучает, что видеоигры развивают, а не чему учат. Уже известно, что видеоигры влияют на умственные способности геймером превосходят людей без игрового опыта в скорости реакции, координации «глаз-рука», точности восприятия, объеме и распределении внимания, пространственном мышлении, счете, переключаемости, когнитивной гибкости и так далее.

Умственные преимущества, которые дарят видеоигры, помогают достигать успеха в реальном мире. Например, тренировка в экшн-видеоигры помогает кадетам летных училищ справляться с управлением самолетами и вертолетами, а студентам-медикам — осваивать хирургические навыки.

Есть также свидетельства, что онлайн-видеоигры учат людей общаться и управлять другими людьми. Иными словами, развивают навыки лидерства. А дети, играющие в стратегические видеоигры, из года в год всё лучше решают задачи.

«Arkanoid» - это аркадная компьютерная игра, она относиться к жанру игр где характерной особенностью является по уровневое дробление игры, когда наградой и целью является право перехода к следующему эпизоду или миссии.

Также для данных игр характерна система набора очков и бонусов (дополнительных наград), предоставляемых за быстроту прохождения, победу над соперником либо нахождение секретных дверей или предметов. Аркадные игры тренируют глазомер, внимание, скорость реакции, но для детей дошкольного возраста рекомендуются ограничения по времени игры.

# **1 ПОСТАНОВКА ЗАДАЧИ**

В качестве задания на производственную технологическую практику требуется разработать компьютерную игру «Arkanoid». Программа разрабатывается для предприятия ООО «ОпенМайГейм».

Постановщикам задачи является руководитель практики от предприятия. Пользователями данной программы являются заинтересованные пользователи.

Игра – тип осмысленной и структурированной непродуктивной деятельности, где мотив лежит не в ее результате, а в самом процессе.

Аркада – жанр компьютерных игр, характеризующийся коротким по времени, но интенсивным игровым процессом.

Главной целью реализации является развитие у игрока моторных функций. Это достигается за счет корректной и сбалансированной настройки управляемых объектов. Управление платформой происходит нажатием на кнопки. Перемещение происходит с определенной скоростью, на определенное расстояние и размер. С помощью платформы игрок отбивает мяч, который летит в заданном направлении и с установленной скоростью. На каждом уровне игроку демонстрируется уникальная фигура, составленная из блоков разных типов, которые игроку потребуется разрушить. За разрушение блоков начисляются очки, а за прохождение уровня начисляются монетки, которые можно потратить в внутриигровом магазине. Так же за разрушения блоков могут выпасть различные бонусы, которые могут как усложнить, так и упростить игру. После завершения игровой компании игрок может играть в бесконечный режим для увеличения рекорда очков.

В разрабатываемой программе будет реализован следующий функционал:

* считывание нажатий клавиш перемещения;
* считывание нажатий кнопок управления интерфейсом;
* вывод данных на экран;
* сохранение данных о последнем запущенном уровне;
* сохранение данных о рекордном количестве очков.

Интерфейс программы должен содержать:

* кнопки для работы с меню игры;
* кнопки для управления платформы;
* кнопки для выбора или подтверждения действия.

Входными данными в программе являются:

* нажатие на кнопки.

Выходными данными в программе являются:

* реакции программы на действия пользователя;
* вывод очков;
* номер уровня;
* количество заработанных денег;
* рекордное количество очков;
* игровые события.

Минимальная конфигурация технического оборудования:

* AMD Athlon™ X2-250 3.0 МГц;
* 1024МБ ОЗУ;
* Видеокарта с объемом памяти не менее 512мб.

Требования к информационной и программной совместимости:

* ОС Windows не ниже версии 7;

Для реализации проекта, полученного как задание на производственную технологическую практику, был выбран игровой движок «Unity5» так как он представляет собой бесплатный мощный игровой движок для разработки мульти платформенных игр с поддержкой всего необходимого функционала. Совместно со средой разработки «Microsoft Visual Studio 2019». При решении поставленной задачи оптимально использовать для представления информационных материалов язык C#, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

# **2 ОПИСАНИЕ ПРОГРАММЫ**

## **2.1 Общие сведения**

Наименование программы: компьютерная игра «Arkanoid».

Автор - Лосев Алесей Сергеевич, группа ПК-42. Дата создания 2020 г.

В качестве игрового движка был выбран Unity. Редактор Unity имеет простой интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе.

Проект в Unity делится на сцены (уровни) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику. В редакторе имеется система наследования объектов;

При компиляции проекта создается исполняемый (.exe) файл игры (для Windows), а в отдельной папке — данные игры.

С 2018 года игровой движок Unity поддерживает только объектно ориентированный язык C#, поэтому именно он был выбран для разработки.

## 

## **2.2 Структура программы**

Наименование и значение файлов хранения данных представлена в рисунке 2.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя файла** | **Тип дынных** | **Размер** | **Описание** |
| RecordFile | Числовой | Целое | Хранит количество очков. |
| LastOpenLavel | Числовой | Целое | Хранит номер последнего открытого уровня. |

Рисунок 2.1 – Наименование и значение

Используемый набор протоколов данных для обмена данными логического типа, между участками, классами и программами. Компьютерная игра «Arkanoid» представляет собой набор сцен, таких как Menu, Game. Каждая из которых содержит в себе различные объекты и компоненты. Сначала пользователь попадает на сцену «Menu», где он может перейти на сцену игры, включить или выключить звук, посмотреть рекордное количество очков. После перехода на сцену «Game» пользователь может преступить к управлению платформой. Отталкиванием мяча платформой и удара его об блоки пользователь получает очки и монеты.

Код программы представлен в приложении А. В проекте имеется сцена «Menu», на которой находятся следующие компоненты:

* AnimationController – содержит методы каждый из которых представляет собой различные анимации;
* SettingsMenu – отвечает за загрузку данных и переключение сцены;
* SettingsMenuItem – класс обозначающий объект сцены «Menu»;

Также в проекте есть сцена «Game» на которой происходит основная часть работы:

* BlockController – класс контролирующий загрузку уровней и урон по блокам;
* EventController – класс, отвечающий за выпадения и обработку событий;
* GameController – контролирует за считывание входных и вывод выходных данных;
* PoolManager – создает некоторое количество выбранного объекта;
* Ball – контролирует поведение мяча;
* Block – контролирует поведение блока;
* Parcer – считывает уровень из файла;
* EventGame – общий класс событий;
* AccelerationBall – ускорение мяча на определенное время;
* PlatformExtension – управление платформой;

## **2.3 Входные и выходные данные**

Организация входных данных представлена в рисунке 2.2.

Запрос на ввод данных

Обработка входных данных, присваивание необходимых параметров, продолжение работы

Рисунок 2.2 – Схема входных данных

Организация выходных данных представлена в рисунке 2.3.

Запрос на вывод данных

Обработка запроса, считывание необходимых параметров, вывод данных, продолжение работы

Рисунок 2.3 – Схема выходных данных

Предоставляется возможность ввода данных, с передачей этих данных в заранее созданной переменной. Форматами входных и выходных данных являются: числовые, символьные, текстовые, логические и вещественные.

## **2.4 Вызов и загрузка**

Вызов программы осуществляется с помощью исполняемого файла «Setup.exe», находящегося на жестком диске, или на переносном носителе. Вход в программу осуществляется только с исполняемого файла «Setup.exe».

# **3 ОПИСАНИЕ ПРИМЕНЕНИЯ**

В ходе производственной технологической практики была разработана компьютерная игра «Arkanoid».

## **3.1 Требования техники безопасности и охраны труда при эксплуатации программы**

При эксплуатации программы необходимо соблюдать правила техники безопасности и охраны труда при работе на ПК:

– продолжительность непрерывной работы на ПК не должна превышать 40 минут с 10 – перерывом.

– общая продолжительность работы за компьютером не должна превышать в день более 4-х часов;

– не допускать попадания внутрь компьютера металлических предметов, воды;

– не допускать чрезмерное накопление пыли внутри компьютера, своевременно проводя работы по техническому обслуживанию;

– не подключать и не отключать интерфейсные кабели при включённом электропитании.

Основной опасностью при работе на ПК является поражение электрическим током, которое может произойти при неисправности ПК или не соблюдении требований техники безопасности. Во избежание опасности поражения электрическим током, обязательными для выполнения являются следующие правила техники безопасности:

– компьютер можно подключить только к электрической розетке;

– не снимать крышку корпуса компьютера при включенном электропитании;

– не работать с компьютером, если повреждён сетевой кабель;

– не касаться частями тела экрана или тыльной стороны монитора стороны монитора, а также питающих проводов;

– не работать на ПК мокрыми руками, в сырой одежде и обуви;

– не работать на ПК во время грозы;

– не пользоваться открытым огнём при работе на ПК;

– вынимать вилку, держаться за вилку, а не за кабель. Не касаться вилки влажными руками;

– полное обесточивание компьютера возможно только при отключении шнура питания от электрической розетки. Поэтому доступ к ней должен быть всегда открыт.

Немедленно отключите компьютер от электрической сети, если:

– появится огонь, дым, необычный звук или запах;

– повреждён сетевой кабель, разъём или розетка;

– внутрь компьютера попала вода или металлический предмет.

Жизненно важным для повышения работоспособности пользователя и сохранения его здоровья, является правильная организация рабочего места пользователя. Кроме того, выбор места важен для надёжной работы компьютера и сопряжённого с ним оборудования. Требования, предъявляемые к организации и оборудованию автоматизированных рабочих мест:

– компьютер предназначен для эксплуатации в закрытых отапливаемых помещений с низкой запыленностью, в котором не должно возникнуть условий для конденсации влаги:

– размеры (площадь, объём) помещения должны соответствовать количеству работающих и размещаемому в них комплексу технических средств;

– планировка рабочего места должна удовлетворять требованиям удобства выполнения работ, экономии энергии и времени, рационального использования производительных площадей и удобства обслуживания устройств ПК, соблюдения охраны и труда;

– при небольшом количестве рабочих предпочтений следует отдавать расстановке ПК у стены, противоположной оконным проёмам;

– конструкция рабочей мебели (столы, кресла и стулья) должна обеспечивать возможность индивидуальной регулировки соответственно росту работающего и создать условия для удобства при работе. Рабочий стол должен регулироваться по высоте в пределах 0.68-0.76 м. Наиболее оптимальным размером рабочей поверхности стола является -1.6×0.9 м. Под ней должно быть свободное пространство для ног с размерами по высоте – не менее 0.6 м от пола, по ширине 0.5 м, по глубине -0.65м;

– в помещении температура воздуха должна содержать 19-21 градусов Цельсия, относительная влажность воздуха 55-56%, скорость движения воздуха не более 0.2м/с;

– уровни звука не должны превышать 50 ДБ. Учитывая, что основным источником шума является принтер и принимая во внимание повсеместное использование практических бесшумных струйных принтеров, этого требования достичь не трудно.

Требования безопасности в аварийных ситуациях.

При перерывах и других нарушениях в подаче электроэнергии компьютер следует выключать. Выключение компьютера обязательно даже в случае отсутствия напряжения в сети питания.

Признаками отключения от нормальной работы ПК являются:

– возникновения необычного шума системного блока или монитора;

– появление запаха гари или озона, возникновения дыма или пламени;

– искажение или исчезновение изображения на мониторе;

– непонятное или исчезновения изображения на мониторе;

– непонятное пользователю неожиданное изменение в работе ПК.

Если при выполнении работы пользователь заменил неисправность, он обязан прекратить работу, сообщить об этом лицу, ответственному за безопасность выполнение работы.

При обнаружении повреждений проводов питания, неисправности заземления и других повреждений электрооборудования, необходимо прекратить работу и сообщить об аварийной ситуации руководителю. Обо всех сбоях в работе оборудования или программного обеспечения необходимо сообщать руководителю.

При появлении рези в глазах, ухудшению видимости, появлении боли в пальцах и кистях рук, усилении сердцебиения необходимо покинуть рабочее место и сообщить о происшедшем руководителю работ. В случае поражения электрическим током работающего рядом необходимо немедленно освободить пострадавшего от действия тока и начать оказание первой помощи, используя соответствующие его состоянию способы: при нарушении дыхания – искусственное дыхание, при нарушении сердечной деятельности – непрямой массаж сердца. Для обесточивания оборудования – выключить автомат питания в помещении. Доложить о несчастном случае руководителю. При необходимости – воспользоваться аптечкой, и вызвать скорую помощь по телефону 103.

Пользователь ПК должен быть обучен правилам тушения пожара, информирован о плане эвакуации людей и имущества из помещения.

В случае возникновения пожара, немедленно вызвать пожарную команду, сообщить о пожаре руководителю и приступить к ликвидации очагов возгорания имеющимися средствами, предварительно обесточив помещение выключением автомата питания или рубильника в коридоре. Лицам, не участвующим в тушении пожара, следует покинуть помещение. При сильном задымлении помещения необходимо закрыть дыхательные пути влажной тканью и выйти из помещения по эвакуационным путям.

## **3.2 Энерго и ресурсосбережение**

Отдельное внимание необходимо уделить энерго и ресурсосбережению. Снизить уровень энерго и ресурсопотребления возможно за счёт наукоёмких энерго- и ресурсосберегающих технологий.

Одним из направлений снижения энергопотребления является использование жидкокристаллических мониторов.

Также по окончанию работы на ПК необходимо полностью выключать и отключать от сети компьютер, монитор, оргтехнику, такую как принтеры, сканеры, ксероксы. В режиме ожидания, по мнению специалистов, компьютер оставлять не следует, так как он потребляет электроэнергию.

Есть ещё много путей, с помощью которых можно снизить энергопотребление. К ним можно отнести такие пути, как использование последних моделей процессоров, чипов и т.д.

# **4 РУКОВОДСТВО ПРОГРАММИСТА**

## **4.1 Назначение программного обеспечения**

В ходе производственной технологической практики была разработана компьютерная игра «Arkanoid». Организационная структура организации представлена в приложении Б.

## **4.2 Характеристики программы**

Минимальная конфигурация технического оборудования:

* AMD Athlon™ X2-250 3.0 МГц;
* 1024МБ ОЗУ;
* Видеокарта с объемом памяти не менее 512мб.
* дисковый накопитель не менее 100 МБ свободного места;
* разрешение экрана должно быть не менее 1280х720.

Необходимое программное обеспечение:

* среда исполнения: ОС от Microsoft Windows 7 или выше (32 – bit / 64 - bit);
* любой доступный базовый редактор текста, поддерживающий формат «.txt».

Контроль правильности выполнения программы осуществляется методом исключения. Программа предотвращает ввод данных неприемлемого формата, путем их отслеживания и исправления до вывода ошибок.

В случае сбоя управления или параметров данных, в программе предусмотрена возможность сохранения.

## **4.3 Установка программы**

За установку программы отвечает инсталляционный пакет «Setup.exe», при загрузке которого сначала будет предложено выбрать язык мастера установки представлено на рисунке 4.1.

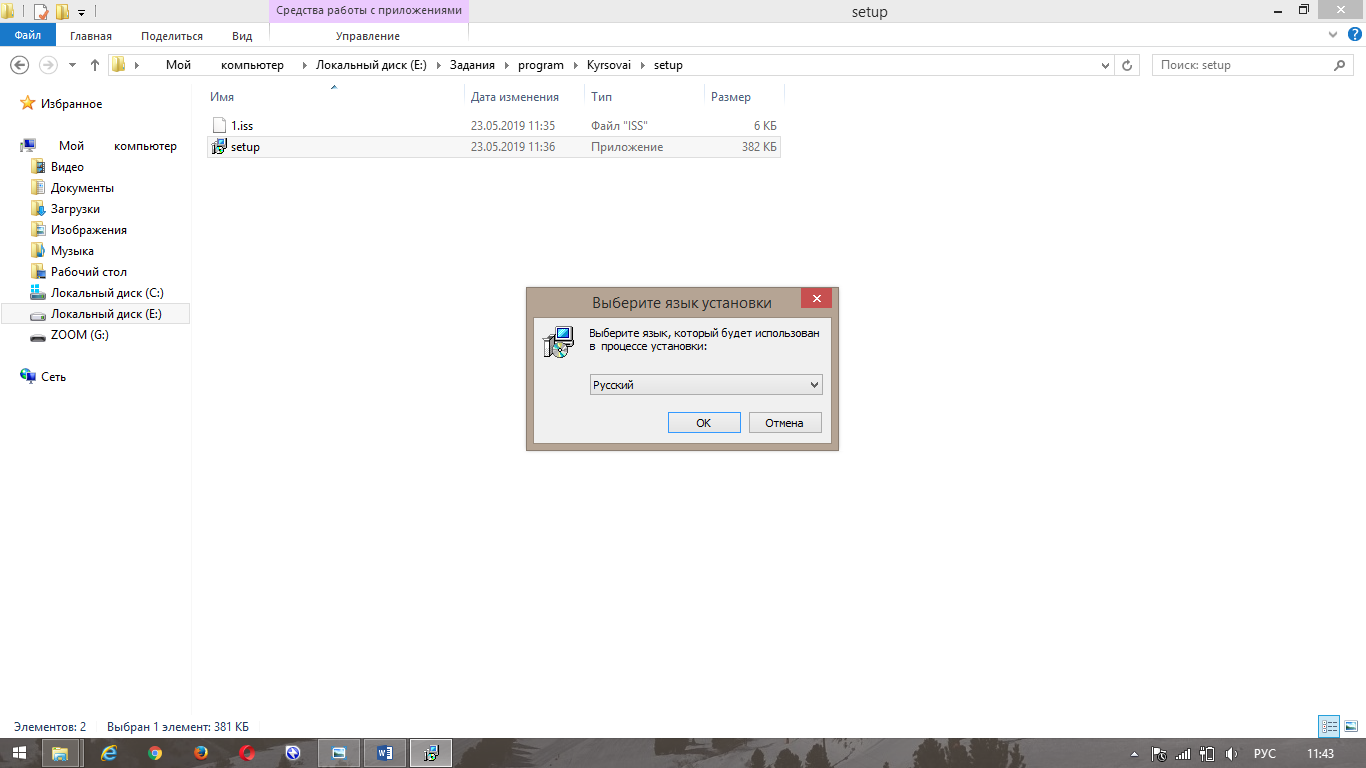


Рисунок 4.1 – Выбор языка мастера установки

После выбора языка откроется мастер, сообщающий о том, какая программа (название, версия) будет установлена на компьютер, как показано на рисунке 4.2.

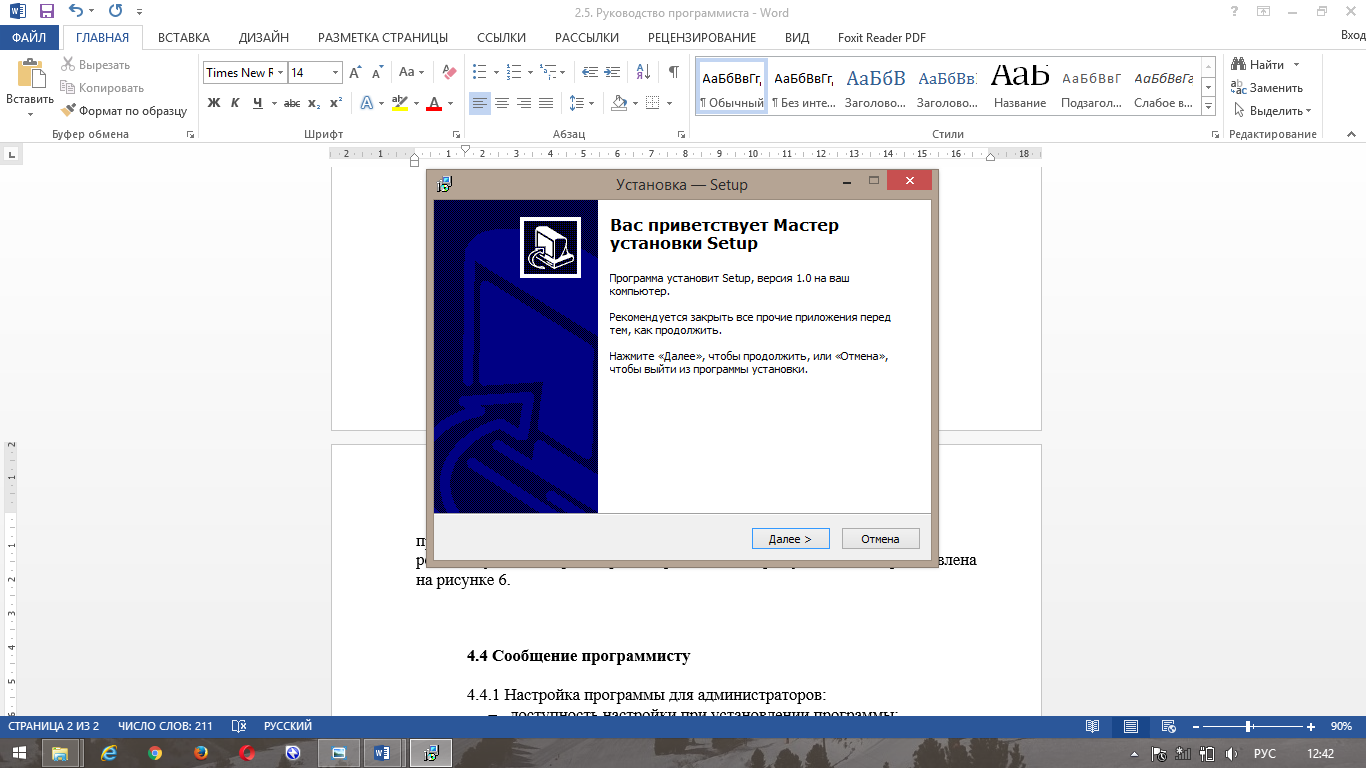


Рисунок 4.2 – Открытие мастера установки

После ознакомления названием и версией программы представляется условия лицензионного соглашения, как показано на рисунке 4.3.

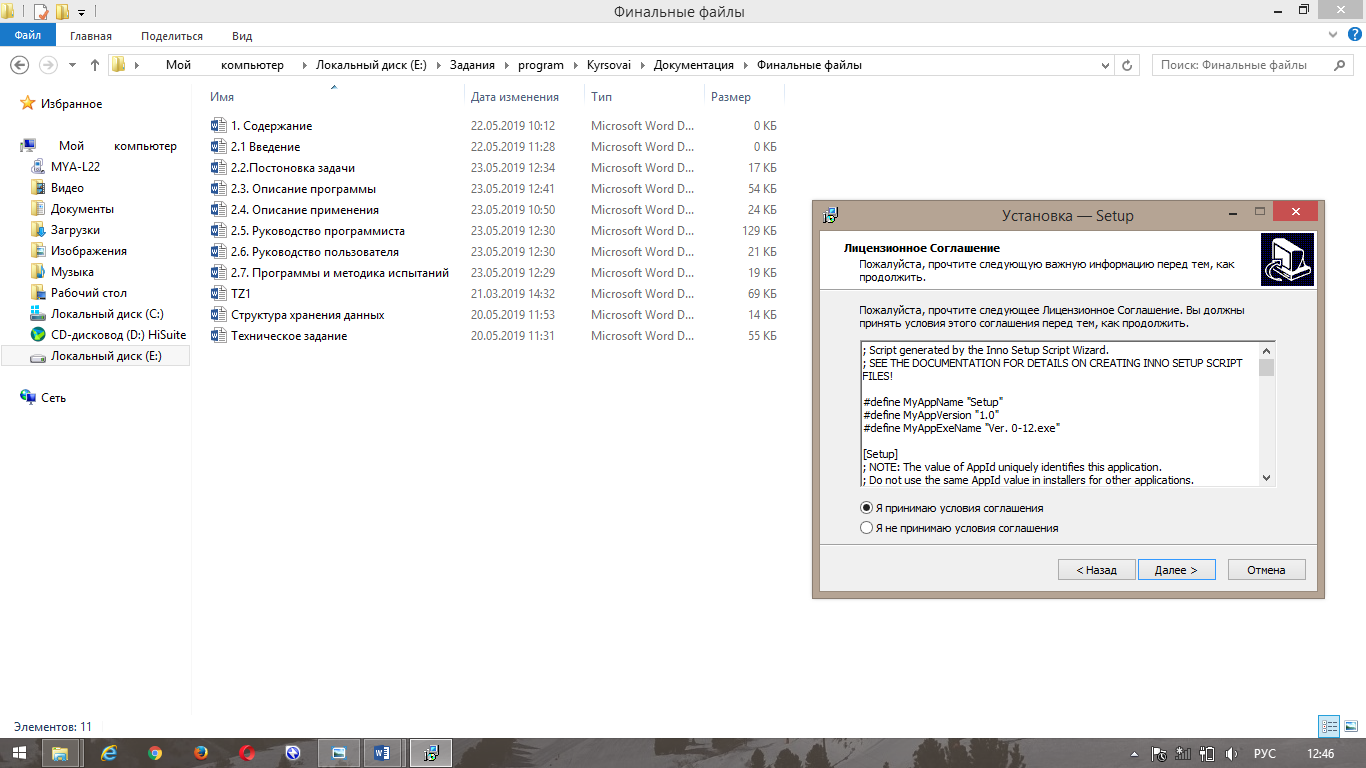


Рисунок 4.3 – Условия лицензионного соглашения

После принятия лицензионного соглашения требуется указать путь, по которому будет установлена программа как показано на рисунке 4.4.

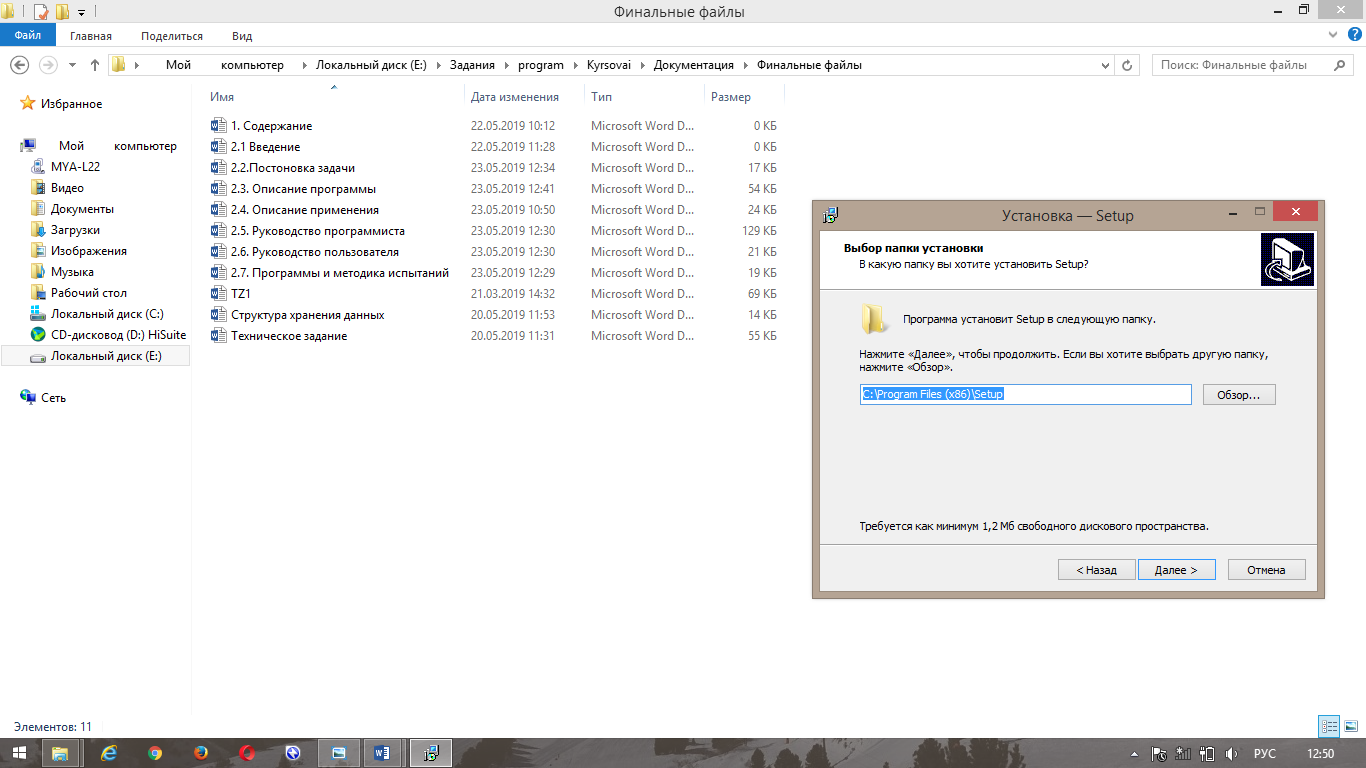


Рисунок 4.4 – Выбор пути установки программы

После установки на ваш компьютер программа будет доступна для запуска, и дальнейшей работы с ней. Для запуска программы необходимо произвести двойное нажатие клавиши мыши по файлу «Arkanoid.exe» расположенном в корневой папке программы или по иконке на рабочем столе.

## **4.4 Сообщения программисту**

Настройка программы для администраторов:

* доступность настройки при установлении программы.

Средства самовосстановления программы не предусмотрены. При удалении одного из комплектующих необходимо переустановить программу.

# **5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

## **5.1 Наименование программы**

В ходе производственной технологической практики была разработана компьютерная игра «Arkanoid».

## **5.2 Условия выполнения программы**

Минимальная конфигурация технического оборудования:

* AMD Athlon™ X2-250 3.0 МГц;
* 1024МБ ОЗУ;
* Видеокарта с объемом памяти не менее 512мб.
* дисковый накопитель не менее 100 МБ свободного места;
* разрешение экрана должно быть не менее 1280х720.

Необходимое программное обеспечение:

* среда исполнения: ОС от Microsoft Windows 7 или выше (32 – bit / 64 - bit);
* любой доступный базовый редактор текста, поддерживающий формат «.txt».

## **5.3 Выполнение программы**

Загрузка программы производится по ярлыку на рабочем столе, или путем запуска файла «Arkanoid.exe» в каталоге с установленной программой. Стартовое меню программы представлено на рисунке 5.1.

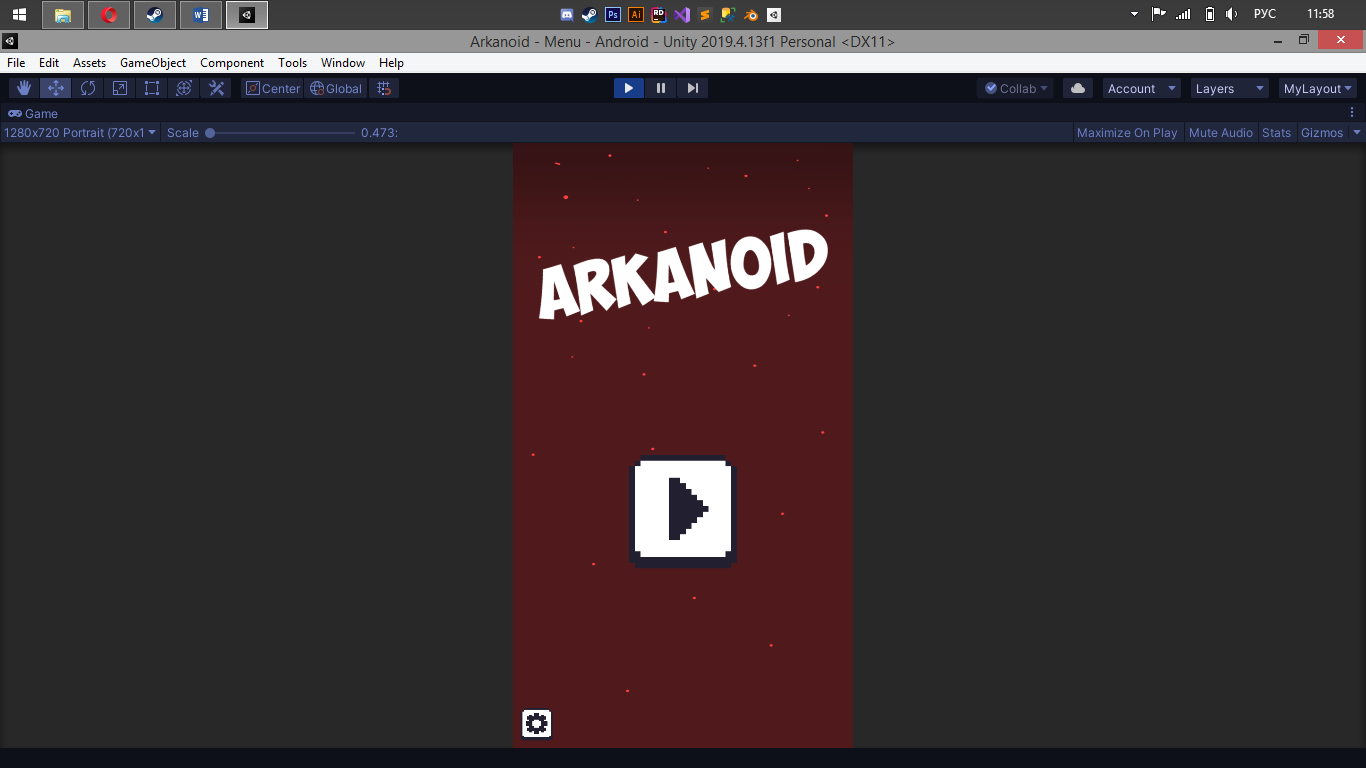


Рисунок 5.1 – Стартовое меню программы

Управление программой осуществляется путем нажатия соответствующих кнопок в меню, отвечающей за определенное действие.

Для открытия панели настроек необходимо нажать на кнопку с изображением шестерёнки представленной на рисунке 5.2.

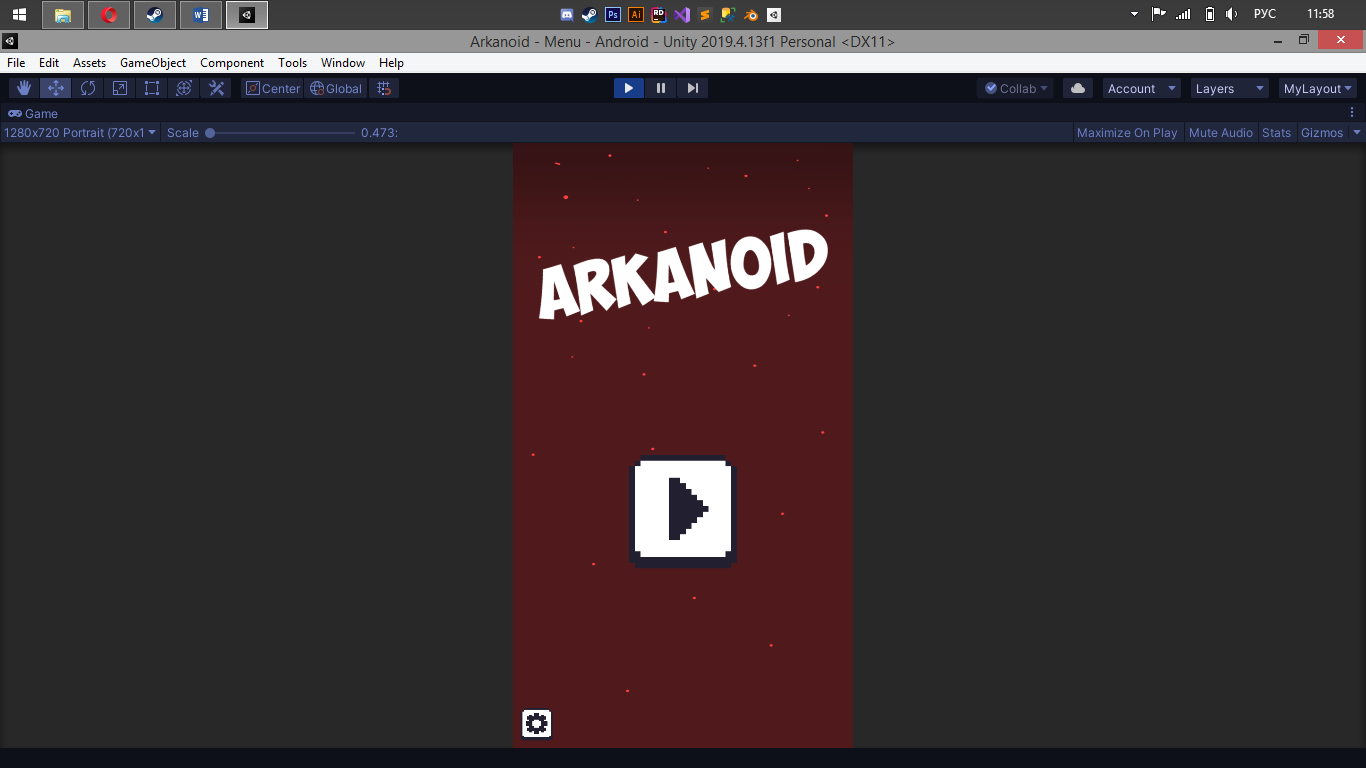


Рисунок 5.2 – Кнопка настроек

Для просмотра максимально набранного количества очков необходимо нажать на кнопку с изображением кубка предоставленном на рисунке 5.3.

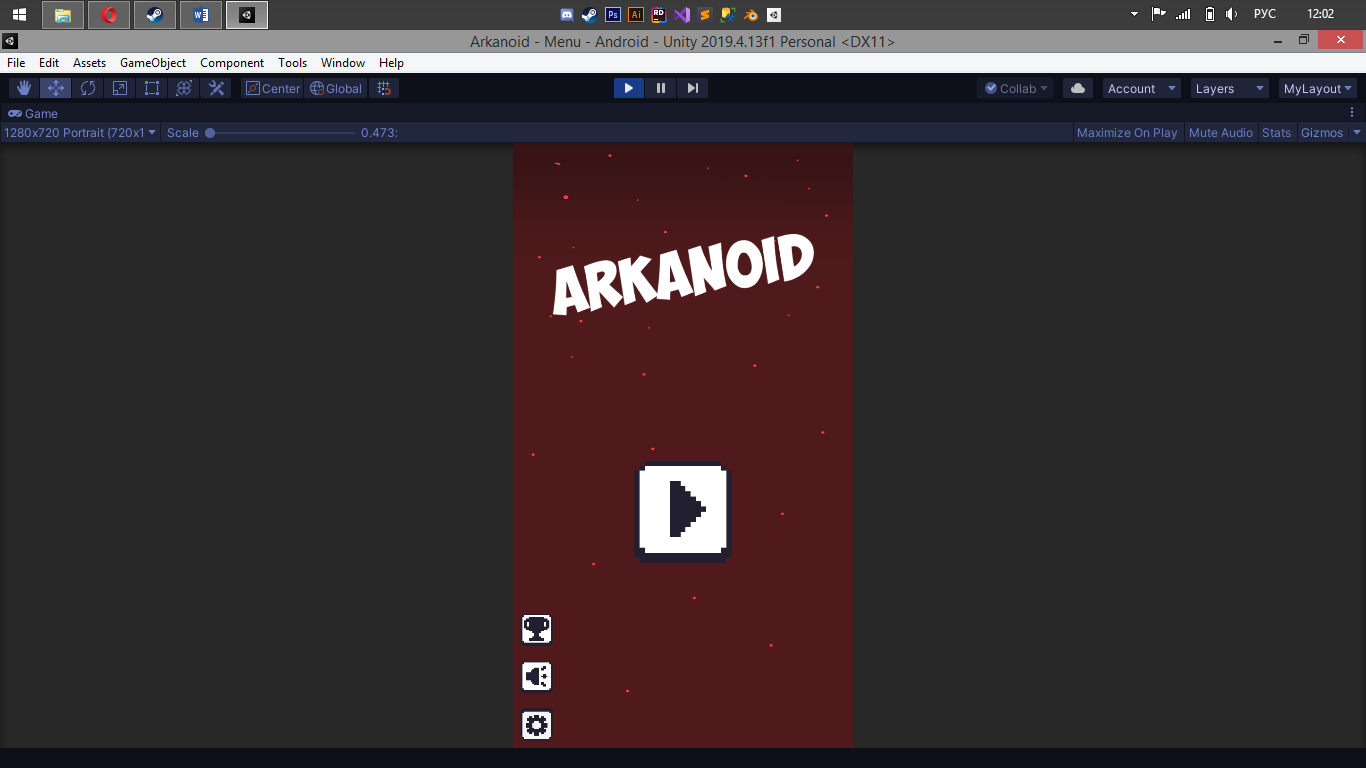


Рисунок 5.3 – Кнопка просмотра рекорда

После нажатия на кнопку просмотра рекорда на экране появиться панель, представленная на рисунке 5.4, для ее закрытия нужно нажать на кнопку, представленную на рисунке 5.3.

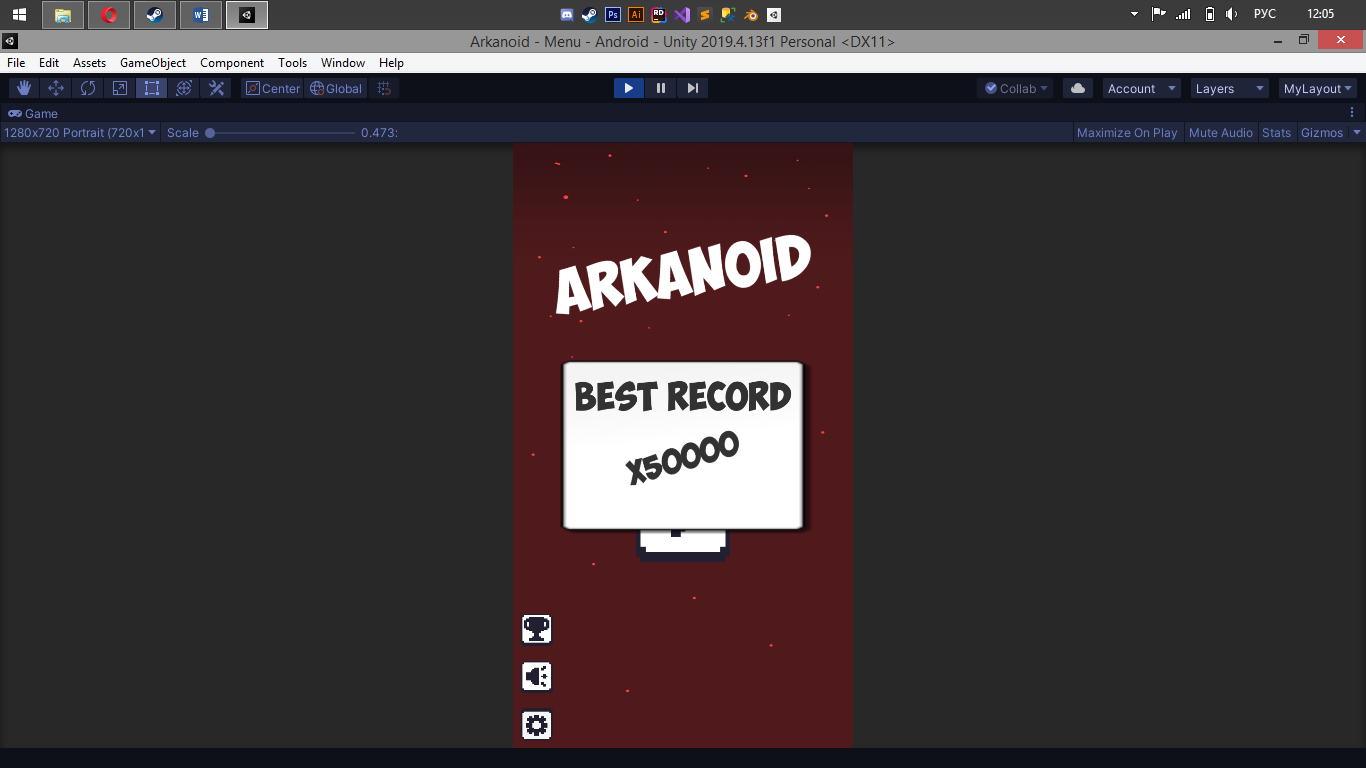


Рисунок 5.4 – Панель рекорда

Для перехода на следующую сцену необходимо нажать на кнопку, представленную на рисунке 5.5.

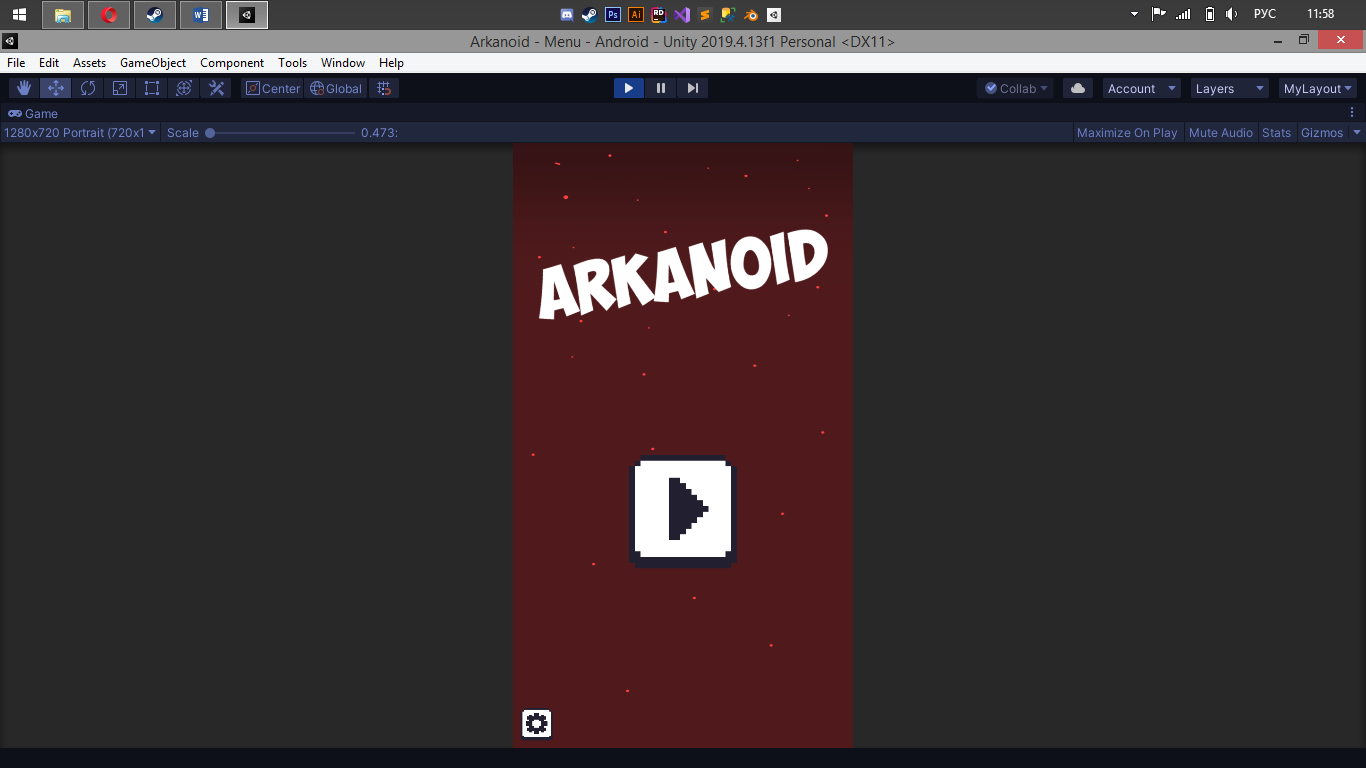


Рисунок 5.5 – Кнопка перехода на следующую сцену

После не продолжительной загрузки откроется основная сцена игры, представленная на рисунке 5.6.

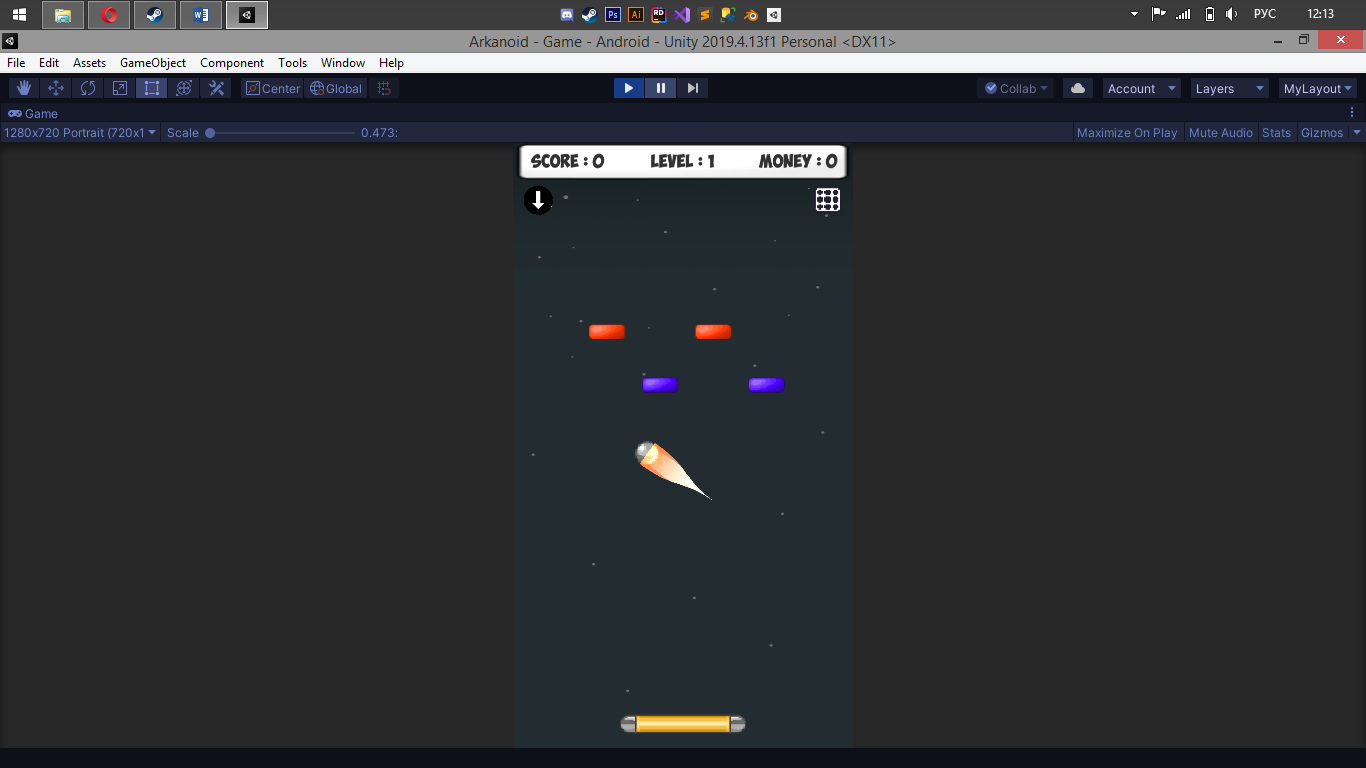


Рисунок 5.6 – Основная сцена

Для управления платформой, представленной на рисунке 5.7 необходимо нажимать на кнопки «Ф» и «В» или же на «стрелка влево», «стрелка в право» на клавиатуре.

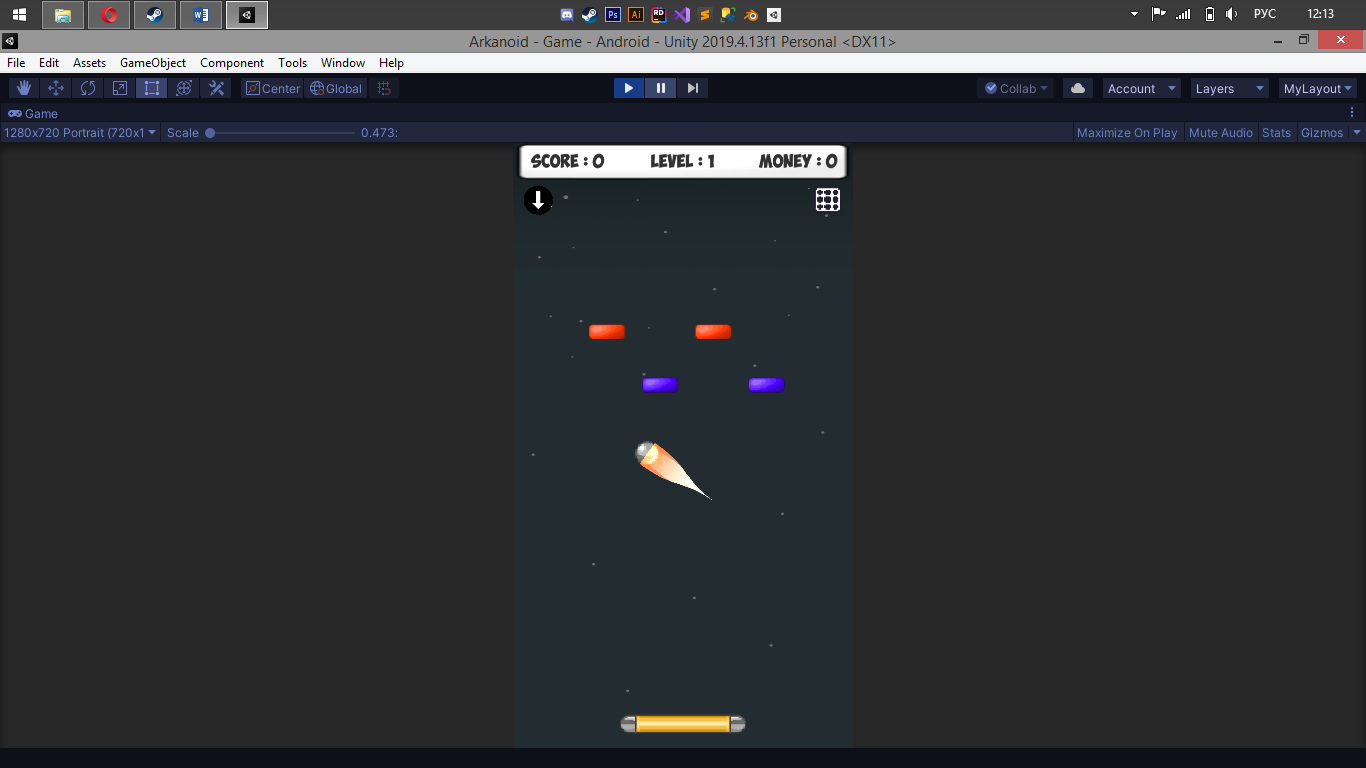


Рисунок 5.7 – Платформа

В случаи застревания мяча или желания изменить направление мяча необходио нажать на кнопку представленную на рисунке 5.8.

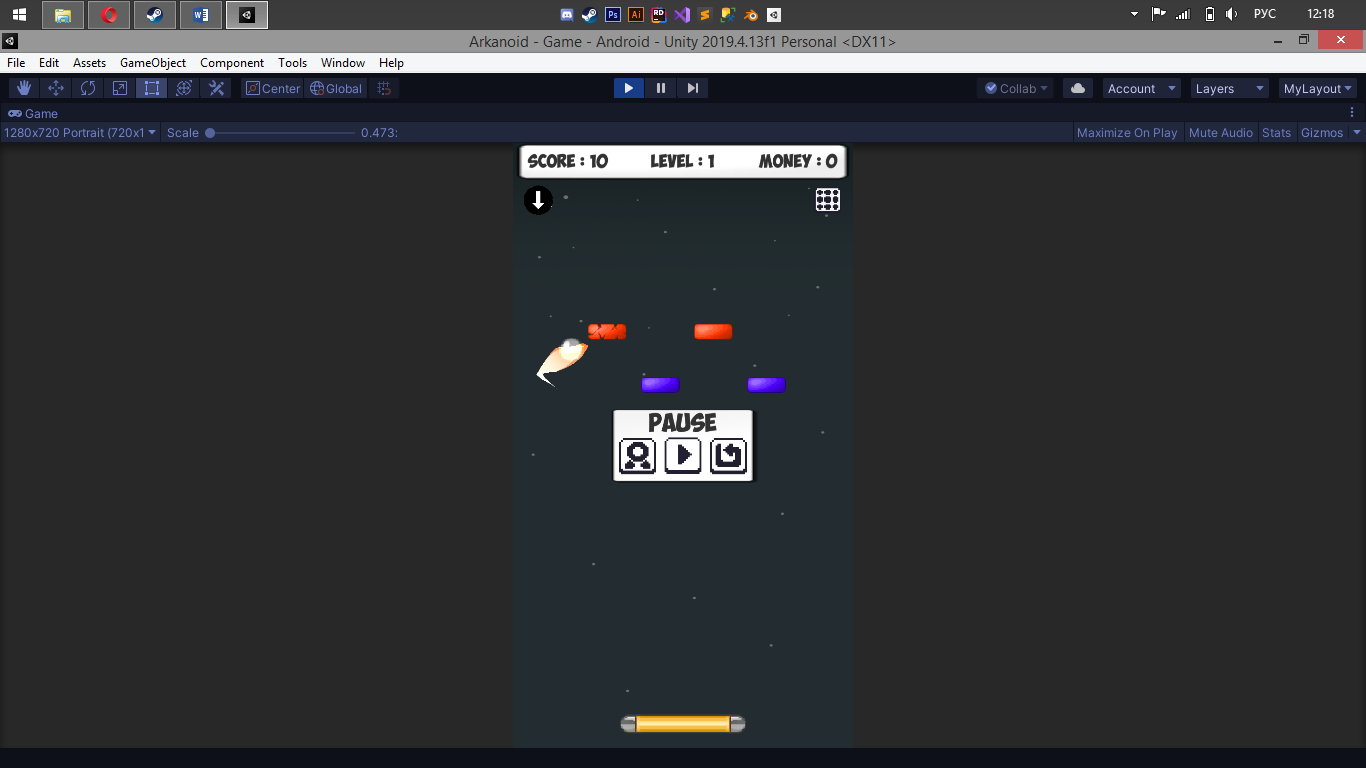


Рисунок 5.8 – Кнопка изменения направления

Для открытия панели меню необходимо нажать на кнопку представленную на рисунке 5.9.

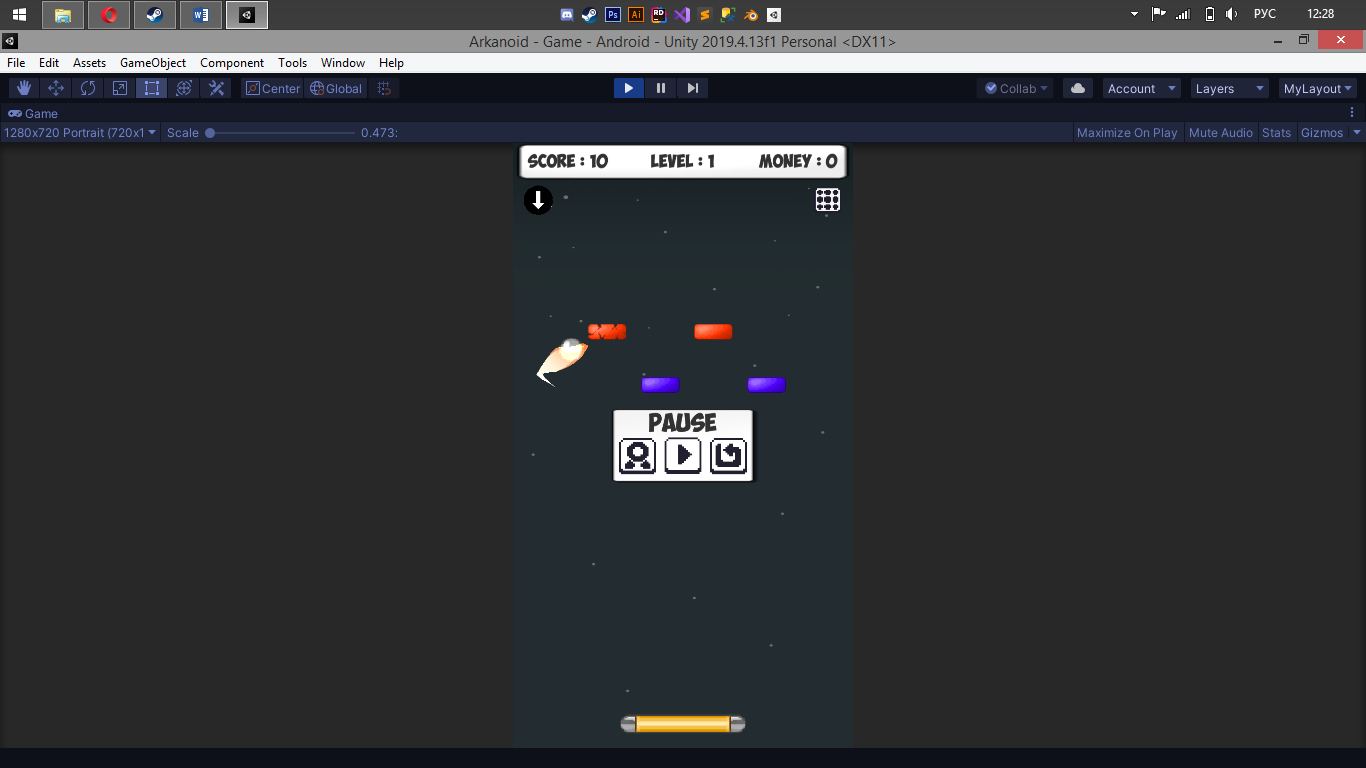


Рисунок 5.9 – Кнопка меню

Поле нажатия на кнопку открытия меню на экране появиться панель предстваленная на рисунке 5.10.

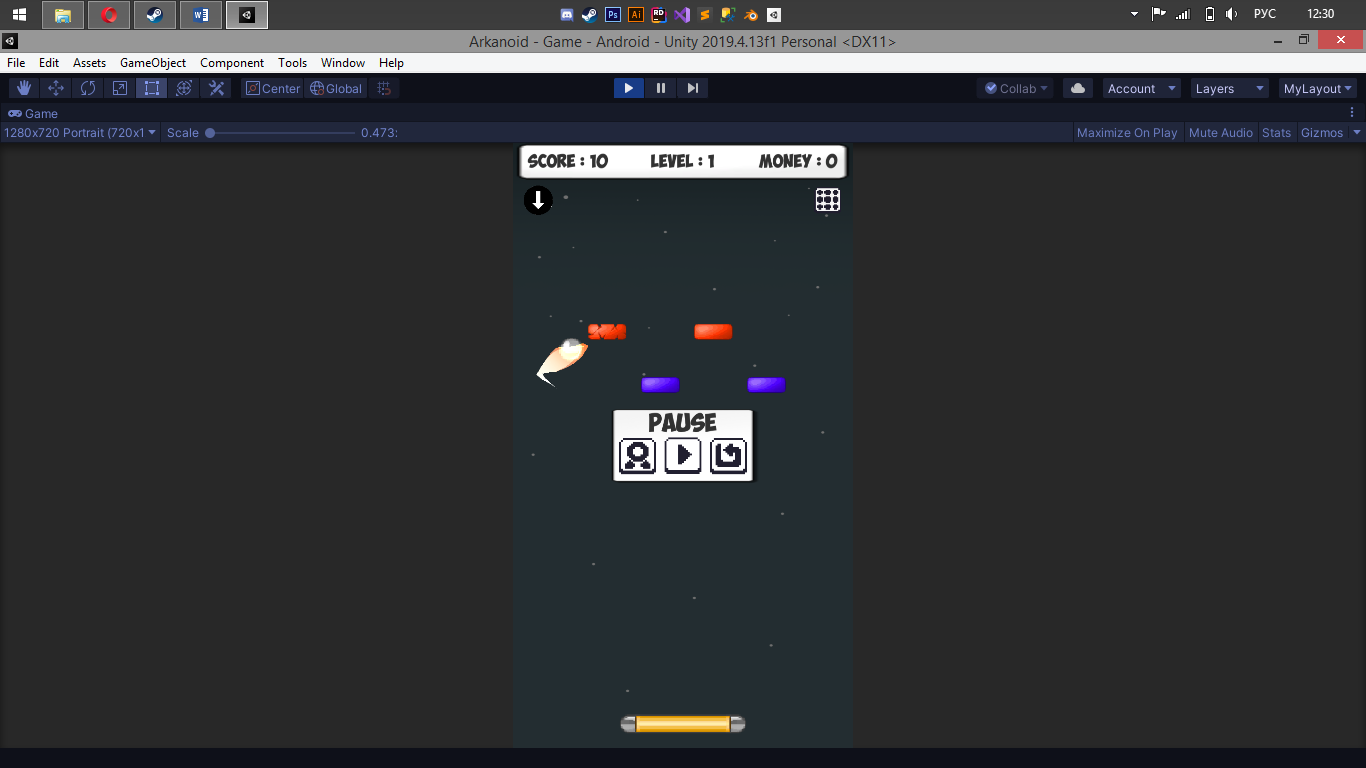


Рисунок 5.10 – Панель меню

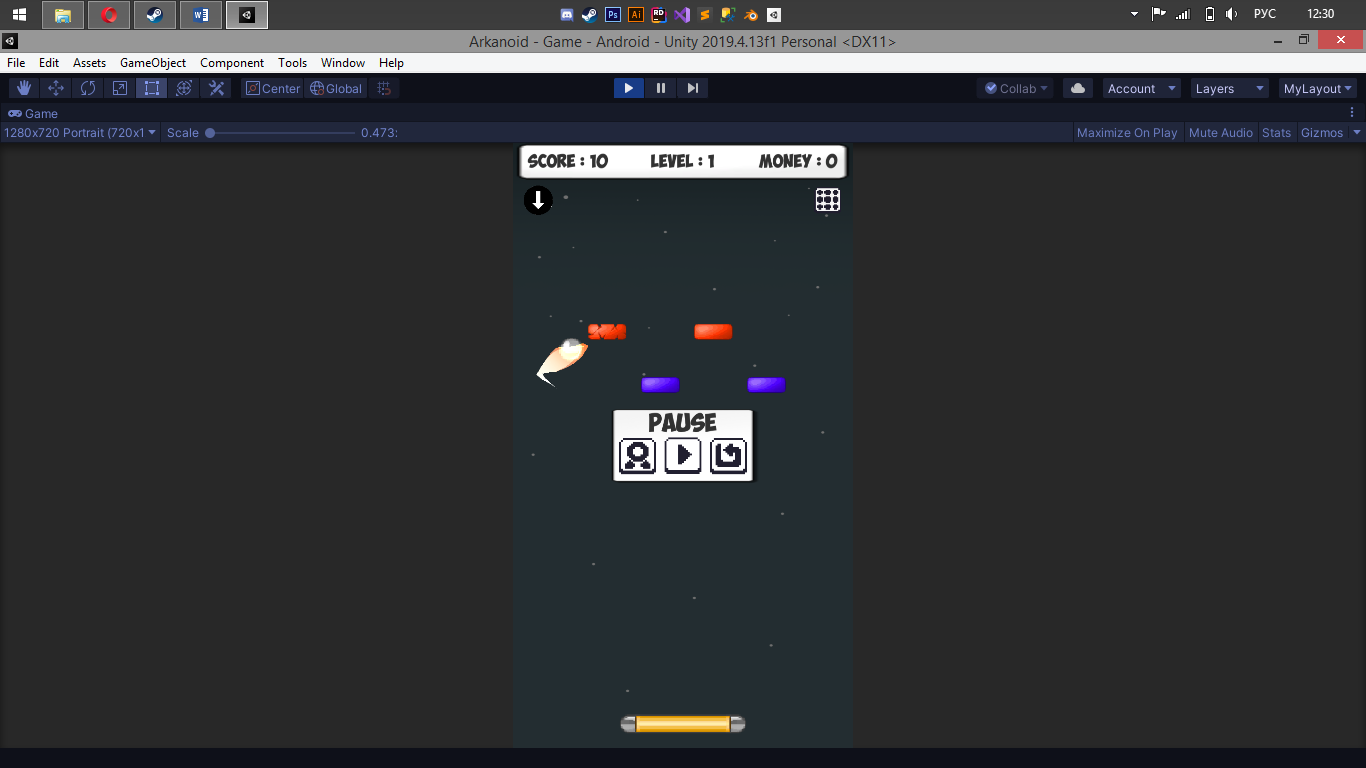
Для продолжения игры необходимо нажать на кнопку представленную на рисунке 5.11.

Рисунок 5.11 – Продолжение игры

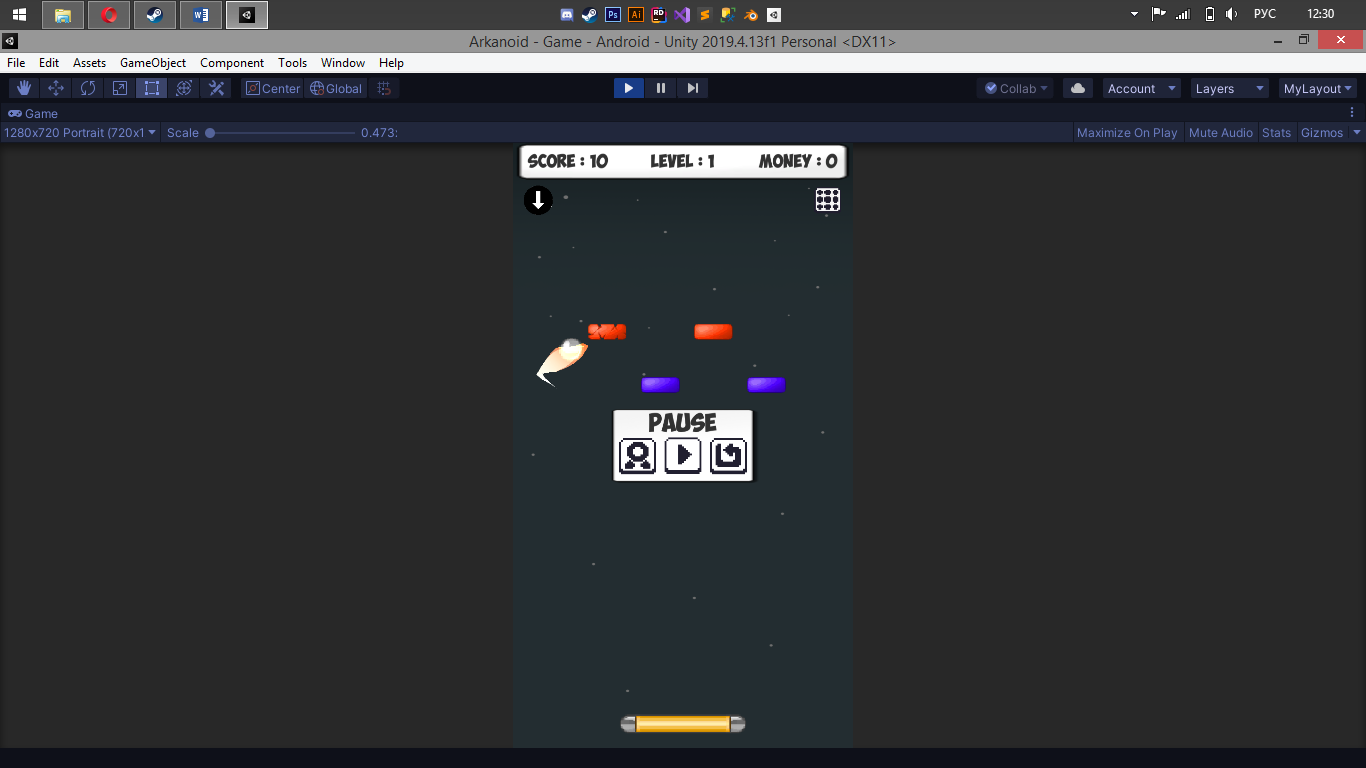
Для перезагрузки уровня необходимо нажать на кнопку представленную на рисунке 5.12.

Рисунок 5.12 – Перезагрузка уровня

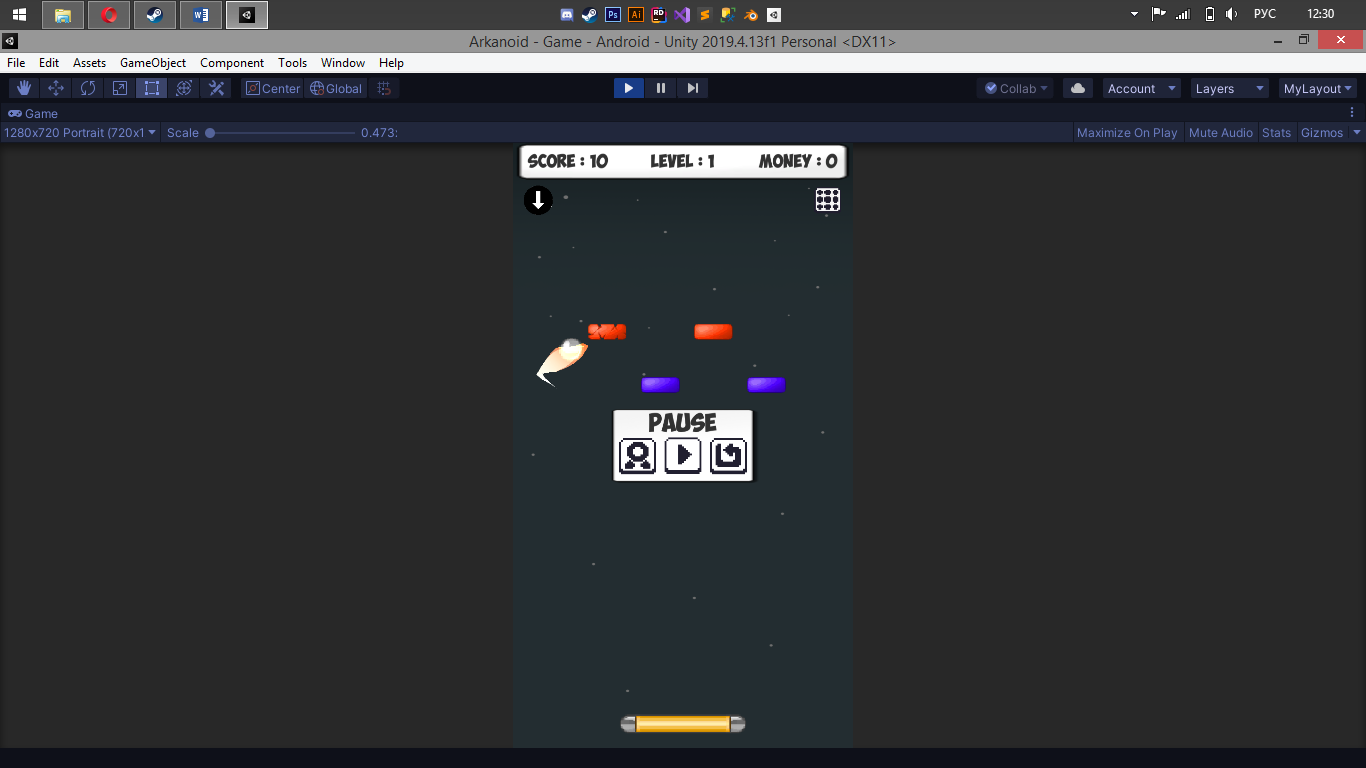
Для возвращения на стартавую сцену необходимо нажать на кнопку представленную на рисунке 5.13.

Рисунок 5.13 – Переход на стартовую сцену

# 

# **6 ПРОГРАММА И ДЕТОДИКА ИСПЫТАНИЙ**

## **6.1 Объект и цель испытаний**

Объектом испытаний является компьютерная игра «Arkanoid».

Целью испытаний является проверка полной работоспособности программы и соответствия её функций требованиям технического задания.

Конфигурация технического оборудования, на котором производились испытания:

* Intel Core™ i3-3217U;
* 8192МБ ОЗУ;
* NVIDIA GeForce GT 720M.
* дисковый накопитель на 750гб;

## **6.2 Порядок испытаний**

Проверка корректности запуска программы

В стандартном запуске программы необходимо подождать загрузки сцены меню, если все настройки были успешно пройдены, то пользователь видит стартовую сцену представлению на рисунке 6.1.

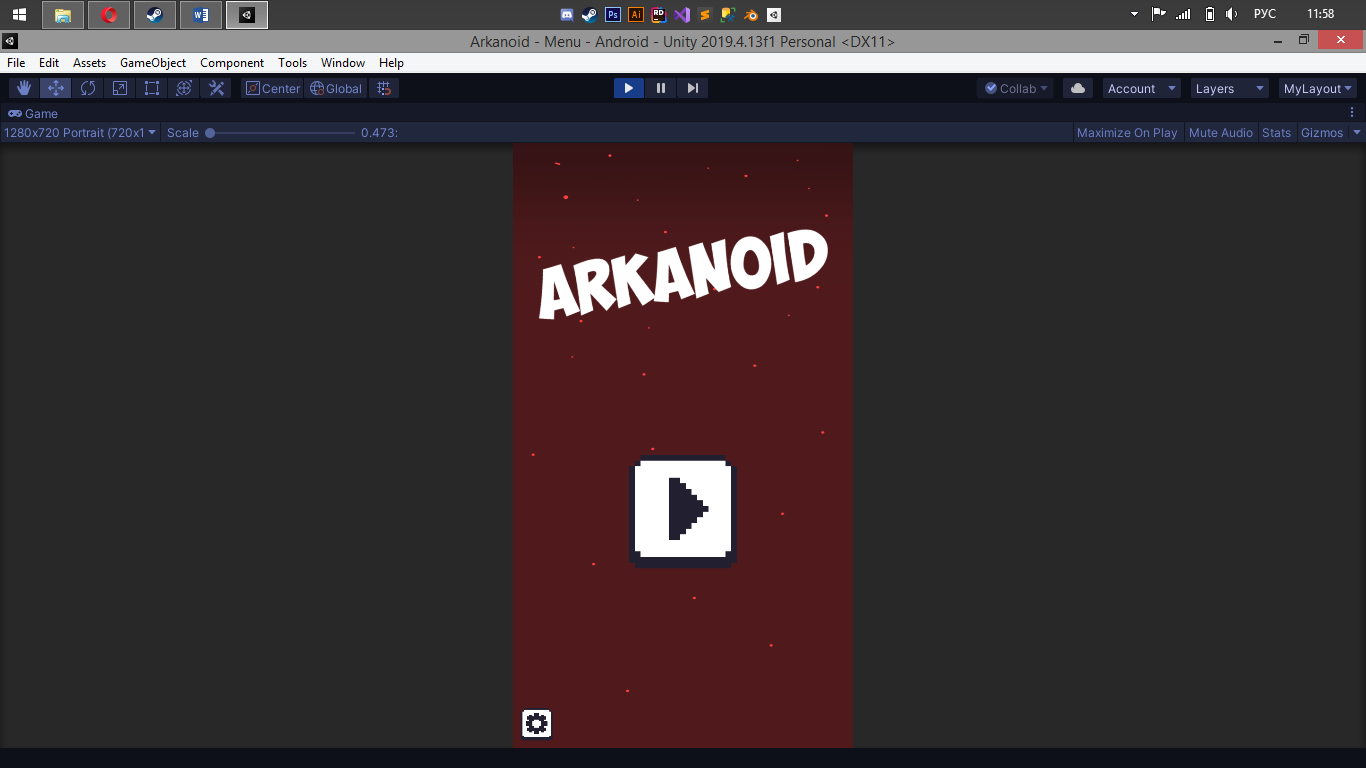


Рисунок 6.1 – Стартовая сцена

После загрузки стартовой сцены необходимо проверить возможность открытия панели настроек, для этого необходимо нажать на кнопку, представленную на рисунке 6.2.

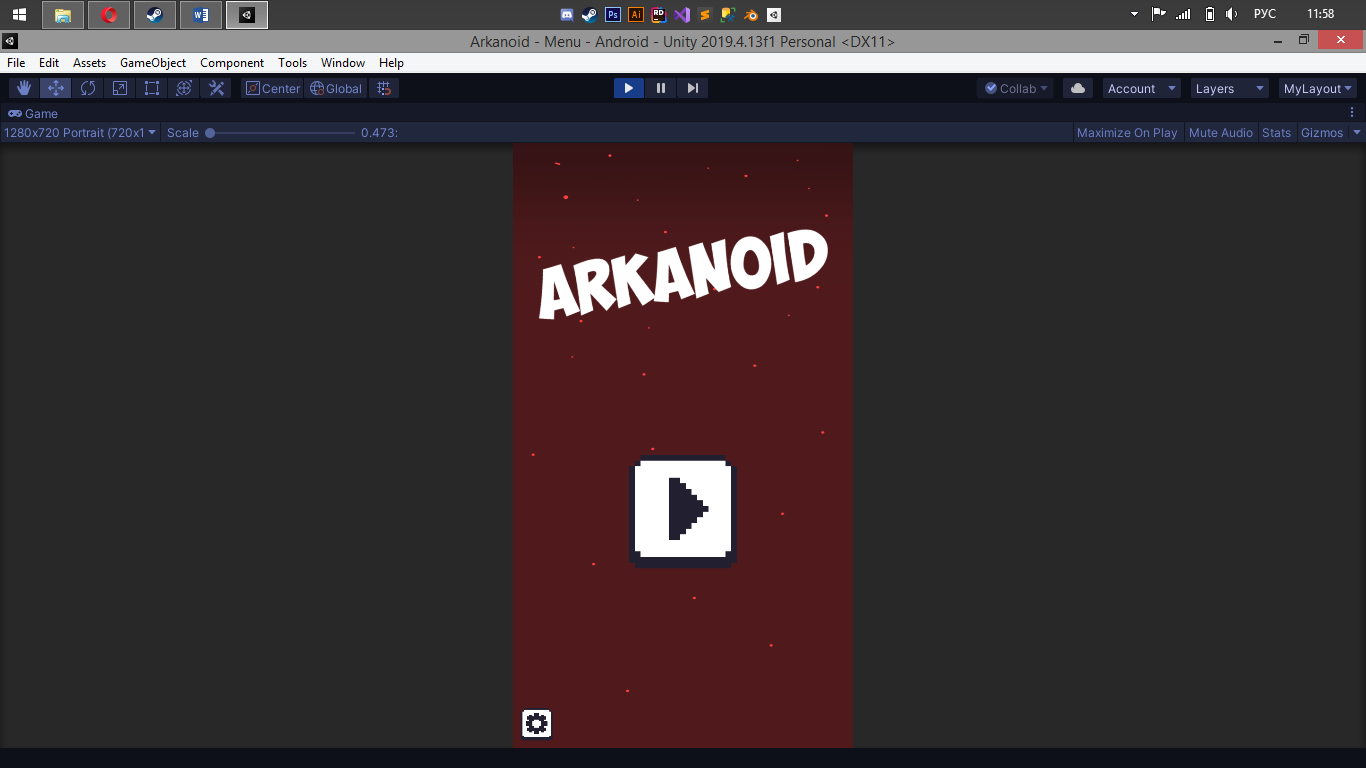


Рисунок 6.2 – Кнопка настроек

После нажатия на кнопку настроек на стартовой сцене должны появиться кнопки, представленные на рисунке 6.3.

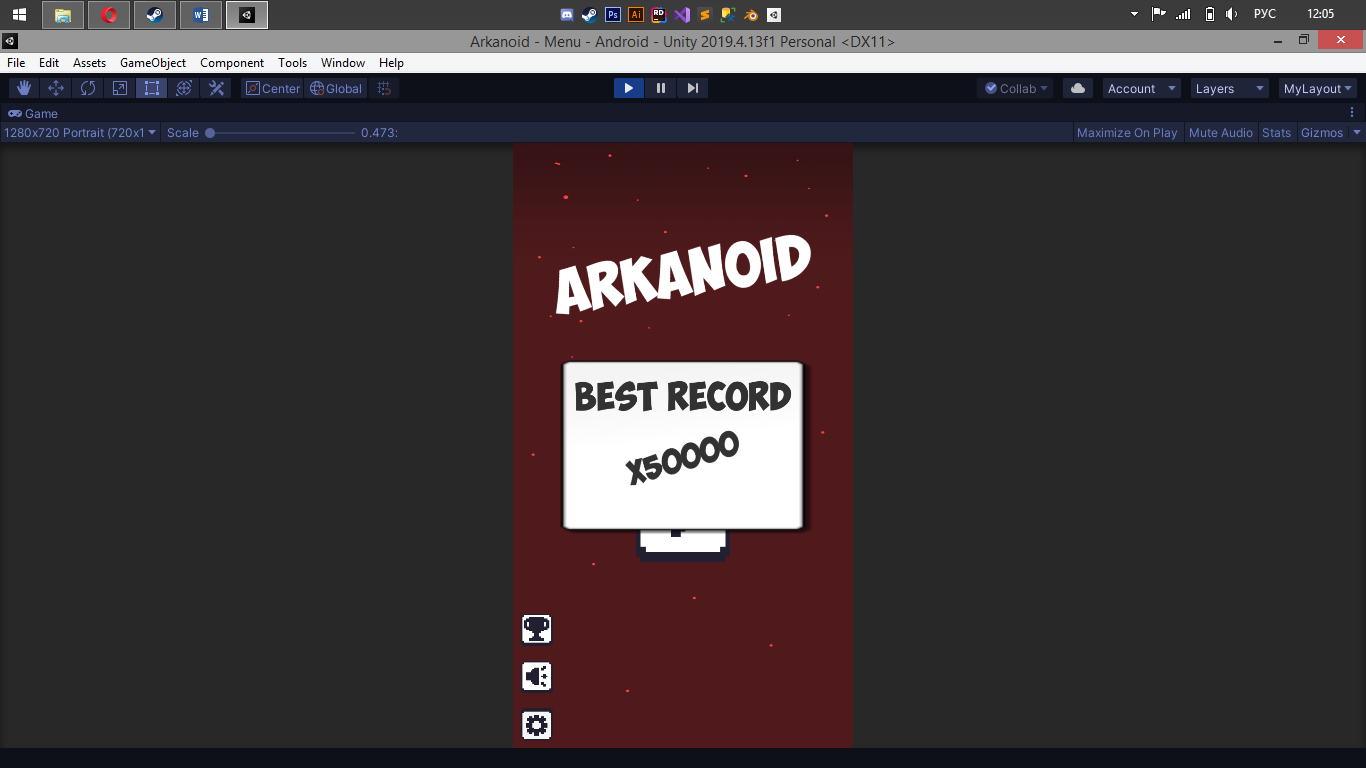


Рисунок 6.3 – Кнопки различных настроек

Для перехода на следующую сцены необходимо нажать на кнопку, представленную на рисунке 6.4.

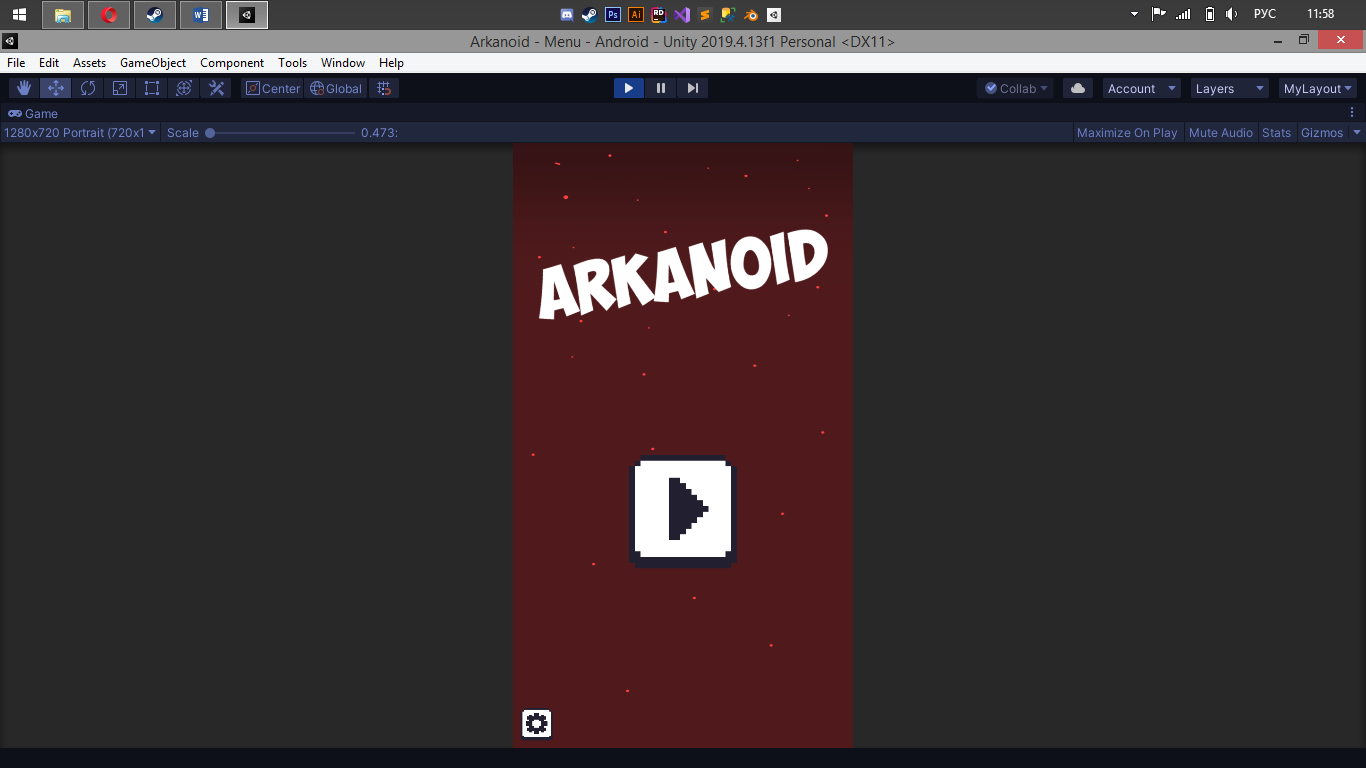


Рисунок 6.4 – Кнопка перехода на следующую сцену

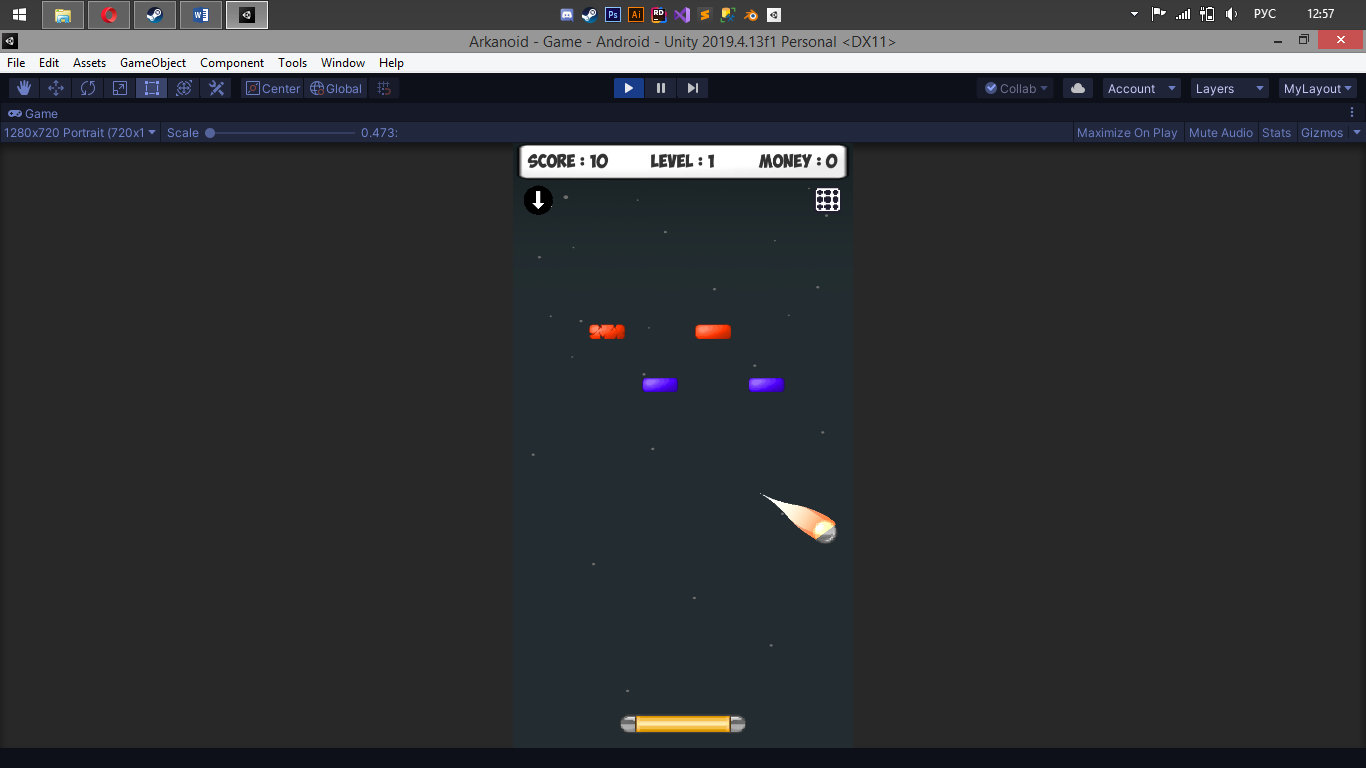
После загрузки на экране появиться основная сцена, представленная на рисунке 6.5.

Рисунок 6.5 – Основная сцена

После нажатия на клавиши управления платформа изменяет свое расположение, перемещение платформы представлено на рисунках 6.6 и 6.7.

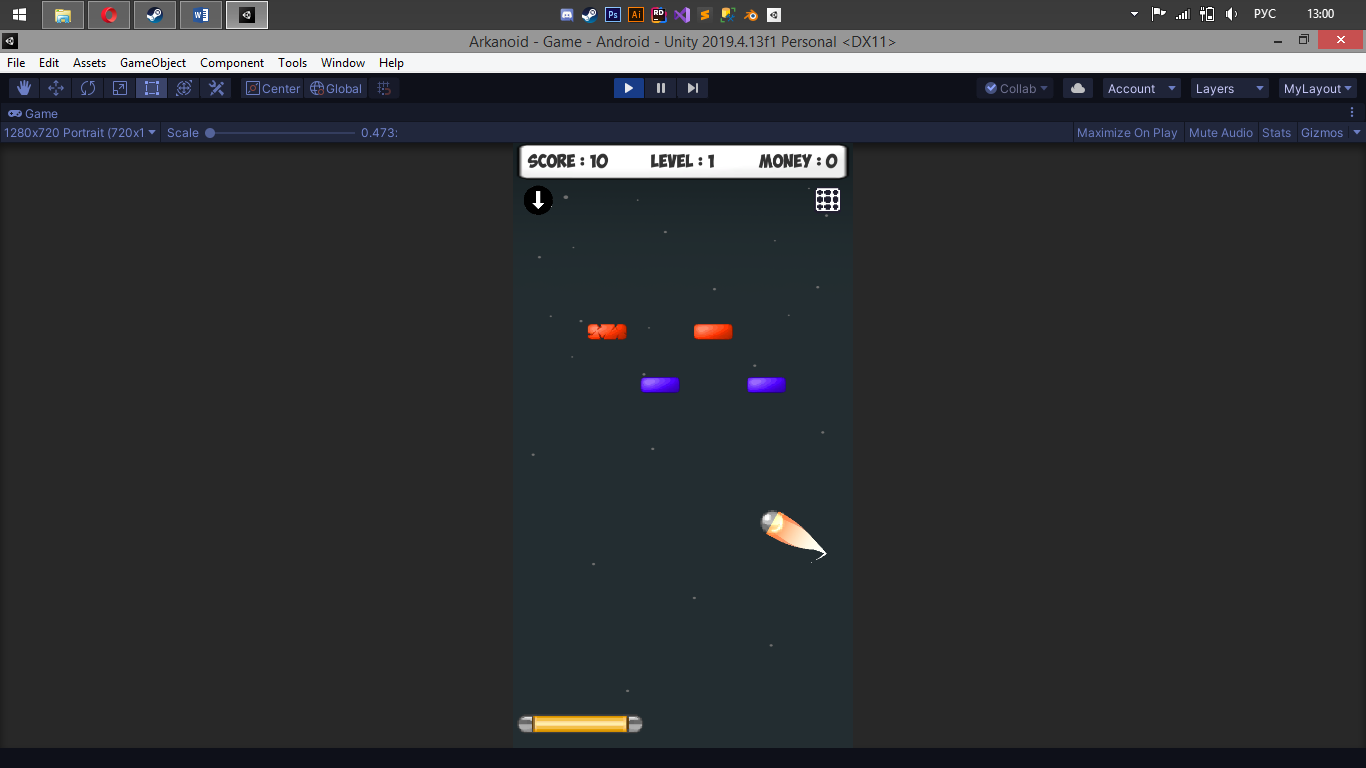


Рисунок 6.6 – Перемещение платформы

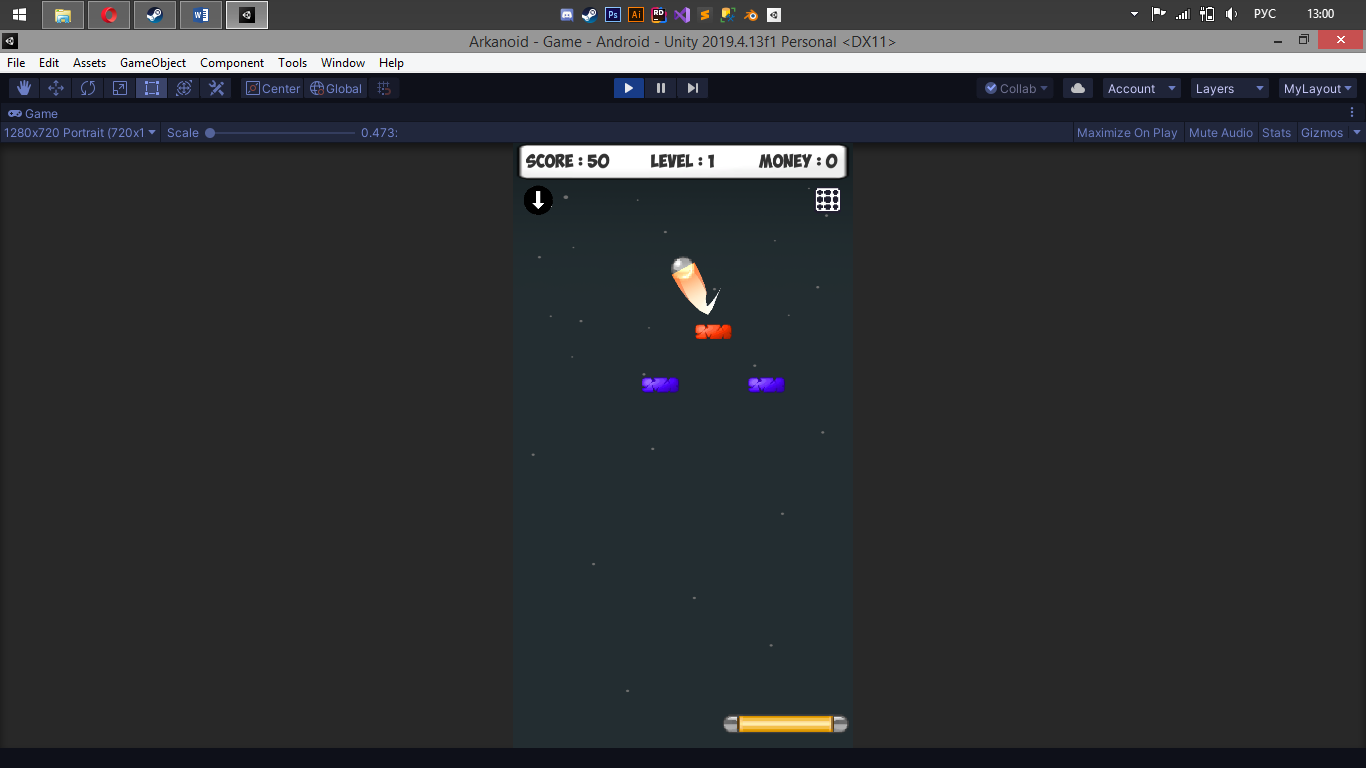


Рисунок 6.7 – Перемещение платформы

После удара мяча по блоку, количество очков должно увеличится, а сам блок исчезнуть, демонстрация этого представлена на рисунках 6.6 и 6.7.

# **ЗАКЛЮЧЕНИЕ**

В ходе производственной технологической практики была разработана компьютерная игра «Arkanoid».

В программе реализованы следующие функции:

* просмотр максимально набранное количество очков;
* переход между сценами;
* перемещение платформы;
* физика полета мяча;
* настройка материалов объектов;
* загрузка уровней;
* уничтожение блоков;
* начисление очков;
* начисление денег;
* изменение направления мяча;
* генерация событий;
* увеличение и уменьшение размера платформы;
* увеличение и замедление скорости мяча;
* возможность расположение на сцене более одного мяча;
* перезагрузка уровней;
* установки игры на паузу.

Программа разработана для непрофессиональных пользователей ПЭВМ, поэтому была максимально упрощена, а интерфейс сделан более простым и понятным.

Все вышеперечисленные качества программы характеризуют ее как программу, точно соответствующую техническому заданию.

В результате создания программы были закреплены умения работы с игровым движком Unity, и знания языка C#, умения реализовать интерфейс, адаптированный под разные разрешения, работа с несколькими сценами, управление поведением множеством количеством объектов, настройка физики объектов, материалов, работа с компонентами как рукописными, так и встроенными.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Виды программ и программных документов. Термины и определения: ГОСТ 19.101-77. – Введ. 01.01.80. – Минск: Межгос. совет по стандартизации, метрологии и сертификации: Белорус. гос. ин-т стандартизации и сертификации, 1980. – 8 с.

2 Единая система программной документации. Общие требования к программным документам. Термины и определения: ГОСТ 19.105-78. – Введ. 01.01.80. – Минск: Межгос. совет по стандартизации, метрологии и сертификации: Белорус. гос. ин-т стандартизации и сертификации, 1980. – 8 с.

3 Единая система программной документации. Требования к программным документам, выполненным печатным способом. Термины и определения: ГОСТ 19.106-78. – Введ. 01.01.80. – Минск: Межгос. совет по стандартизации, метрологии и сертификации: Белорус. гос. ин-т стандартизации и сертификации, 1980. – 16 с.

4 Единая система программной документации. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества. Термины и определения: ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации: Белорус. гос. ин-т стандартизации и сертификации, 2001. – 16 с.

5 Единая система программной документации. Описание программы. Требования к содержанию, оформлению и контролю качества. Термины и определения: ГОСТ 19.402-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации: Белорус. гос. ин-т стандартизации и сертификации, 2001. – 20 с.

6 Единая система программной документации. Описание применения. Требования к содержанию и оформлению. Термины и определения: ГОСТ 19.502-78. – Введ. 01.01.1980. – Минск: Межгос. совет по стандартизации, метрологии и сертификации: Белорус. гос. ин-т стандартизации и сертификации, 2001. – 4 с.

7 Чебыкин, Р. Самоучитель C# и C++. Современные технологии / Р. Чебыкин. – М.: Мир, 2013. – 624 с.

8 Unity для разработчика. Мобильные мультиплатформенные игры  
Мэннинг Д., Батфилд-Эддисон П.

# 

# **ПРИЛОЖЕНИЕ А**

**Код программы**

using Arkanoid.Utils;

using UnityEngine;

using DG.Tweening;

public class AnimationController : Singleton<AnimationController>

{

[SerializeField] float timeAnimation = 0f;

[SerializeField] float scaleAnimation = 0f;

[SerializeField] LoopType loopType;

[SerializeField] Ease ease;

private void Awake()

{

DontDestroyOnLoad(gameObject);

}

public void ResetAnimation(RectTransform rectTransform)

{

rectTransform = GetComponent<RectTransform> ();

rectTransform.DOScale (new Vector3 (scaleAnimation, scaleAnimation, 0), timeAnimation).SetLoops (-1, loopType).SetEase (ease);

}

public void AnimationPulsation(RectTransform rectTransform)

{

rectTransform.DOKill ();

rectTransform.localScale = new Vector3 (1f, 1f, 1f);

rectTransform.DOScale (new Vector3 (scaleAnimation, scaleAnimation, 0), timeAnimation).SetLoops (-1, loopType).SetEase (ease);

}

public void AnimationPulsation(Transform rectTransform)

{

rectTransform.DOKill ();

rectTransform.localScale = new Vector3 (1f, 1f, 1f);

rectTransform.DOScale (new Vector3 (scaleAnimation, scaleAnimation, 0), timeAnimation).SetLoops (-1, loopType).SetEase (ease);

}

}

=======================

using System.Collections;

using System.Collections.Generic;

using Arkanoid.Utils;

using UnityEngine;

public class BallController : Singleton<BallController>

{

[SerializeField] private int numberBall = 10;

[SerializeField] private GameObject ballPrefab;

private List<GameObject> balls;

public int countActiveBalls;

public delegate void OnGame(Transform target);

public event OnGame onDamageBlock;

private void Awake()

{

CreateBalls();

}

private void CreateBalls()

{

balls = new List<GameObject>();

balls = PoolManager.Instance.GetObjects(ballPrefab, numberBall, gameObject.transform);

}

private void CreateBall()

{

balls.Add(PoolManager.Instance.GetObject(ballPrefab, gameObject.transform));

}

public GameObject GetBall()

{

countActiveBalls++;

foreach (var ball in balls)

{

if (!ball.activeSelf)

{

ball.SetActive(true);

ball.transform.position = ballPrefab.transform.position;

ball.GetComponent<Ball>().Move();

return ball;

}

}

CreateBall();

var newBall = balls[balls.Count-1];

newBall.SetActive(true);

return newBall;

}

public GameObject GetActiveBall()

{

//countActiveBalls++;

foreach (var ball in balls)

{

if (ball.activeSelf)

{

return ball;

}

}

return null;

}

public void HideAllBalls()

{

foreach (var ball in balls)

{

ball.SetActive(false);

}

countActiveBalls = 0;

}

public void CheckLose()

{

--countActiveBalls;

if (countActiveBalls <= 0)

{

GameController.Instance.LoseGame();

}

}

public void DamageControll(Transform target)

{

GameController.Instance.UpDateCoins();

onDamageBlock?.Invoke(target);

}

}

==================

using System;

using System.Collections.Generic;

using Arkanoid.Utils;

using UnityEngine;

using Random = UnityEngine.Random;

public class BlockController : Singleton<BlockController>

{

public int numberBlocks;

[HideInInspector ]public int countActiveBlocks;

public GameObject blocks;

public SpriteBlock[] spriteBlocks;

private List<GameObject> \_poolBlocks;

[Serializable]

public struct SpriteBlock

{

public Sprite fullXpSprite;

public Sprite damageXpSprite;

}

private void Awake()

{

CreateBlocks();

}

private void CreateBlocks()

{

\_poolBlocks = new List<GameObject>();

\_poolBlocks = PoolManager.Instance.GetObjects(blocks, numberBlocks, gameObject.transform);

}

public Sprite[] GetSprite(int index)

{

Sprite[] sprites = new Sprite[2];

sprites[0] = spriteBlocks[index].fullXpSprite;

sprites[1] = spriteBlocks[index].damageXpSprite;

return sprites;

}

public Block GetBlock()

{

foreach (var obj in \_poolBlocks)

{

if (!obj.activeSelf)

{

var block = obj.GetComponent<Block>();

block.boxCollider2D = block.GetComponent<BoxCollider2D>();

AnimationController.Instance.AnimationPulsation(block.transform);

obj.SetActive(true);

return block;

}

}

return null;

}

public void HideAllBlocks()

{

foreach (var block in \_poolBlocks)

{

block.transform.SetParent(gameObject.transform);

//block.GetComponent<Block>().ResetSettings();

block.SetActive(false);

}

}

public void CheckWin()

{

--countActiveBlocks;

if (countActiveBlocks <= 0)

{

GameController.Instance.WinGame();

}

}

}

=================

using System;

using System.Collections.Generic;

using UnityEngine;

using Random = UnityEngine.Random;

namespace Controllers

{

public class EventController : MonoBehaviour

{

public static EventController Instance;

[SerializeField] EventObject objectEvent;

[SerializeField] private int percentageDrop = 10;

private const int MaxPercent = 100;

[SerializeField] public List<Events> Eventses;

[Serializable]

public struct Events

{

public int PercentMin;

public int PercentMax;

public EventGame EventGame;

}

private void Awake()

{

if (Instance == null)

{

Instance = this;

}

else

{

Destroy(this);

}

}

void Start()

{

OnSubscription();

}

private void OnSubscription()

{

BallController.Instance.onDamageBlock += CreateEventObject;

}

private void OnDestroy()

{

BallController.Instance.onDamageBlock -= CreateEventObject;

}

private void CreateEventObject(Transform target)

{

int percent = Random.Range(0, MaxPercent);

if (percent <= percentageDrop)

{

var obj = Instantiate(objectEvent);

obj.transform.position = target.position;

}

}

public void EventGeneration()

{

int percent = Random.Range(0, MaxPercent);

foreach (var \_event in Eventses)

{

//Debug.Log("Percent : " + percent);

if (percent >= \_event.PercentMin && percent <=\_event.PercentMax)

{

\_event.EventGame.Play();

}

}

}

}

}

==================

using System;

using Arkanoid.Utils;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class GameController : Singleton<GameController>

{

[SerializeField] private int multipleForMoney = 100;

[SerializeField] private int multipleForBoll = 1000;

public Platform platformPrefab;

public Transform gridObjects;

public Transform borders;

[HideInInspector]public Platform platform;

private Ball \_ball;

private int score;

private int money;

[Header("UI")]

public GameObject panelLose;

public GameObject panelPause;

public Text coinsText;

public Text moneyText;

public Text levelText;

public int numberLevel;

float offsetObjects = -0.775f;

const int pointsPerHit = 10;

const float sizeRow = 170f;

const float pixelPerUnit = 100f;

const float posBlocks = 1f;

private Block[,] \_blocks;

void Start()

{

score = 0;

money = 0;

CreateScene();

}

private void HidePanels()

{

panelLose.SetActive(false);

panelPause.SetActive(false);

}

private void CreateScene()

{

UpDateText(score, money);

platform = Instantiate(platformPrefab);

float ratio = (float) Screen.height/Screen.width;

float ortSize = ratio / 20f;

borders.localScale = new Vector2(1 - ortSize, 1);

BallController.Instance.GetBall();

CreateBlocks();

}

private void CreateBlocks()

{

var modelLevel = Parcer.Instance().LoadModels(numberLevel);

\_blocks = new Block[modelLevel.size.x, modelLevel.size.y];

for (int j = 0; j < \_blocks.GetLength(1); j++)

{

for (int i = 0; i < \_blocks.GetLength(0); i++)

{

Block obj = \_blocks[i, j];

if (obj == null)

{

obj = BlockController.Instance.GetBlock();

obj.boxCollider2D.isTrigger = true;

obj.transform.parent = gridObjects;

}

obj.x = i;

obj.y = j;

obj.transform.localPosition = new Vector2(obj.x\*(sizeRow/pixelPerUnit + offsetObjects), obj.y\*(sizeRow/pixelPerUnit + offsetObjects));

obj.transform.localScale = new Vector3(1,1,1);

\_blocks[obj.x, obj.y] = obj;

}

}

foreach (Parcer.BlockObject block in modelLevel.blocks)

{

Block obj = \_blocks[block.x, block.y];

obj.type = (Block.EBlockType) Enum.Parse(typeof (Block.EBlockType), block.type, true);

obj.boxCollider2D.isTrigger = false;

obj.spriteRender = obj.GetComponent<SpriteRenderer>();

obj.InstallSettings();

BlockController.Instance.countActiveBlocks++;

}

float width = (modelLevel.size.x + 2)\*(sizeRow + offsetObjects\*pixelPerUnit);

float height = (modelLevel.size.y)\*(sizeRow + offsetObjects\*pixelPerUnit);

float screenHeight = Camera.main.orthographicSize \* 2 \* pixelPerUnit;

float screenWidth = screenHeight \* Camera.main.aspect;

float koef = screenWidth < width ? screenWidth/width : 1;

gridObjects.localScale = new Vector2(koef, koef);

float newKoef = screenHeight < height\*koef ? screenHeight/height\*koef : 1;

gridObjects.localScale = new Vector2(koef\*newKoef\*0.95f, koef\*newKoef\*0.95f);

width -= (sizeRow + offsetObjects\*pixelPerUnit);

gridObjects.localPosition = new Vector3(

-width/pixelPerUnit/3.5f\*gridObjects.localScale.x,

0,

0);

gridObjects.localPosition = new Vector2(gridObjects.localPosition.x, posBlocks);

}

public void UpDateText(int score, int money)

{

coinsText.text = "Score : " + score;

moneyText.text = "Money : " + money;

levelText.text = "Level : " + numberLevel;

}

public void UpDateCoins()

{

score += pointsPerHit;

if ((score % multipleForMoney) == 0)

{

money += 1;

}

if ((score % multipleForBoll) == 0)

{

BallController.Instance.GetBall();

}

UpDateText(score, money);

}

public void LoseGame()

{

panelLose.SetActive(true);

SetPaused(true);

}

public void WinGame()

{

++numberLevel;

RestartGame();

}

public void PauseGame(bool paused)

{

panelPause.SetActive(paused);

SetPaused(paused);

}

public void ExitGame()

{

SetPaused(false);

SceneManager.LoadScene(0);

}

public void RestartGame()

{

BlockController.Instance.HideAllBlocks();

BlockController.Instance.countActiveBlocks = 0;

BallController.Instance.HideAllBalls();

Destroy(platform.gameObject);

score = 0;

//money = 0;

SetPaused(false);

HidePanels();

CreateScene();

}

public void StuckBall()

{

if (money >= 1)

{

money--;

UpDateText(score, money);

BallController.Instance.HideAllBalls();

BallController.Instance.GetBall();

}

}

public void SetPaused(bool paused)

{

if (paused)

{

Time.timeScale = 0.0f;

}

else

{

Time.timeScale = 1.0f;

}

}

}

=====================

using System.Collections;

using UnityEngine;

using UnityEngine.UI;

using DG.Tweening;

using UnityEngine.SceneManagement;

public class SettingsMenu : MonoBehaviour

{

[Header("UI")]

[SerializeField] private Text nameGameText;

[SerializeField] private Button playButton;

[SerializeField] private Image LoadingImg;

[SerializeField] private Image LoadingImg2;

[SerializeField] private Text progressText;

[Space] [Header("Space detwwen menu items")] [SerializeField]

private Vector2 spacing;

[Space] [Header("Main button rotation")]

[SerializeField] private float rotationDuration;

[SerializeField] private Ease rotationEase;

[Space] [Header("Animation")]

[SerializeField] private float expendDuration;

[SerializeField] private float collapseDuration;

[SerializeField] private Ease expendEase;

[SerializeField] private Ease collapseEase;

[Space] [Header("Fading")]

[SerializeField] private float expendFadeDuration;

[SerializeField] private float collapseFadeDuration;

private Button mainButtom;

private SettingsMenuItem[] menuItems;

private bool isExpanded = false;

private Vector2 mainButtomPosition;

private int itemsCount;

private void Start()

{

AnimationController.Instance.AnimationPulsation(nameGameText.rectTransform);

AnimationController.Instance.AnimationPulsation(playButton.GetComponent<RectTransform>());

itemsCount = transform.childCount - 1;

menuItems = new SettingsMenuItem[itemsCount];

for (int i = 0; i < itemsCount; i++)

{

menuItems[i] = transform.GetChild(i + 1).GetComponent<SettingsMenuItem>();

}

mainButtom = transform.GetChild(0).GetComponent<Button>();

mainButtom.onClick.AddListener(ToggleManu);

mainButtom.transform.SetAsLastSibling();

mainButtomPosition = mainButtom.transform.position;

ResetPositions();

}

void ResetPositions()

{

for (int i = 0; i < itemsCount; i++)

{

menuItems[i].trans.position = mainButtomPosition;

}

}

void ToggleManu()

{

isExpanded = !isExpanded;

if (isExpanded)

{

for (int i = 0; i < itemsCount; i++)

{

//menuItems[i].trans.position = mainButtomPosition + spacing \* (i + 1);

menuItems[i].trans.DOMove(mainButtomPosition + spacing \* (i + 1), expendDuration).SetEase(expendEase);

menuItems[i].img.DOFade(1f, expendFadeDuration).From(0f);

}

}

else

{

for (int i = 0; i < itemsCount; i++)

{

//menuItems[i].trans.position = mainButtomPosition;

menuItems[i].trans.DOMove(mainButtomPosition, collapseDuration).SetEase(collapseEase);

menuItems[i].img.DOFade(0f, collapseFadeDuration);

}

}

mainButtom.transform

.DORotate(Vector3.forward \* 180f, rotationDuration)

.From(Vector3.zero)

.SetEase(rotationEase);

}

private void OnDestroy()

{

mainButtom.onClick.RemoveListener(ToggleManu);

}

public void ButtonPlay(int sceneID)

{

LoadingImg.gameObject.SetActive(true);

StartCoroutine(AsyncLoad(sceneID));

}

IEnumerator AsyncLoad(int sceneID)

{

AsyncOperation operation = SceneManager.LoadSceneAsync (sceneID);

while (!operation.isDone)

{

float progress = operation.progress / 0.9f;

LoadingImg2.fillAmount = progress;

progressText.text = string.Format("{0:0}%", progress \* 100);

yield return null;

}

}

}

====================

using System;

using UnityEngine;

using Random = UnityEngine.Random;

public class Ball : MonoBehaviour

{

public float minSpeed;

public float maxSpeed;

public int damage = 1;

void Start()

{

Move();

}

public void Move()

{

gameObject.GetComponent<Rigidbody2D>().velocity = new Vector2(Random.Range(minSpeed, maxSpeed),

Random.Range(minSpeed, maxSpeed));

}

public void Move(bool isSlowly, float kof)

{

var velocity = gameObject.GetComponent<Rigidbody2D>().velocity;

if(isSlowly) gameObject.GetComponent<Rigidbody2D>().velocity = velocity / kof;

else gameObject.GetComponent<Rigidbody2D>().velocity = velocity \* kof;

}

private void OnCollisionEnter2D(Collision2D other)

{

if (other.gameObject.GetComponent<Block>())

{

other.gameObject.GetComponent<Block>().RegistredDamage(damage);

BallController.Instance.DamageControll(gameObject.transform);

}

else if (other.gameObject.GetComponent<Border>() && other.gameObject.GetComponent<Border>().isLose)

{

gameObject.SetActive(false);

BallController.Instance.CheckLose();

}

}

}

=================

using UnityEngine;

public class Block : MonoBehaviour

{

[SerializeField] public SpriteRenderer spriteRender = null;

[SerializeField] public BoxCollider2D boxCollider2D = null;

[SerializeField] public Sprite fullXp;

[SerializeField] public Sprite damageXp;

[SerializeField] public EBlockType type = EBlockType.Empty;

[SerializeField] private int primaryXp = 2;

private int Xp;

public int x = 0;

public int y = 0;

public enum EBlockType

{

Yellow = 0,

Red,

Blue,

Pink,

Orange,

Green,

Empty

}

void Start()

{

//ResetSettings();

}

public void InstallSettings()

{

int index = 0;

switch (type)

{

case EBlockType.Yellow: index = 0;

break;

case EBlockType.Red: index = 1;

break;

case EBlockType.Blue: index = 2;

break;

case EBlockType.Pink: index = 3;

break;

case EBlockType.Orange: index = 4;

break;

case EBlockType.Green: index = 5;

break;

}

Sprite[] sprites = BlockController.Instance.GetSprite(index);

fullXp = sprites[0];

damageXp = sprites[1];

Xp = primaryXp;

spriteRender.sprite = fullXp;

}

public void RegistredDamage(int damage)

{

Xp -= damage;

spriteRender.sprite = damageXp;

if(Xp <= 0)

{

gameObject.SetActive(false);

spriteRender.sprite = null;

BlockController.Instance.CheckWin();;

//Destroy(gameObject);

}

}

}

=================

using UnityEngine;

using SimpleJSON;

public class Parcer

{

public class BlockObject

{

public int x;

public int y;

public string type;

}

public class Size

{

public int x;

public int y;

}

public class LevelInfo

{

public int level\_id;

}

public class ModelJson

{

public Size size;

public LevelInfo levelInfo;

public BlockObject[] blocks;

}

static Parcer \_instance = null;

public static Parcer Instance()

{

return \_instance ?? (\_instance = new Parcer());

}

public const int CountLevels = 5;

public ModelJson LoadModels(int level)

{

if (level > CountLevels)

{

level = 1;

}

var path = "Levels/Level\_" + level;

var asset = Resources.Load<TextAsset>(path);

if(asset == null) return null;

var json = asset.text;

var newJson = JSON.Parse(json);

var model = new ModelJson

{

size = JsonUtility.FromJson<Size>(newJson["size"].ToString()),

levelInfo = JsonUtility.FromJson<LevelInfo>(newJson["level\_info"].ToString()),

};

model.blocks = new BlockObject[newJson["board"].Count];

for(var i = 0; i < newJson["board"].Count; i++)

{

model.blocks[i] = JsonUtility.FromJson<BlockObject>(newJson["board"][i].ToString());

}

return model;

}

}

==============

using System;

using UnityEngine;

[Serializable]

public class EventGame : MonoBehaviour

{

public delegate void OnEvent();

public event OnEvent onEvent;

public void Play()

{

onEvent?.Invoke();

}

}

=================

using System.Collections;

using UnityEngine;

public class AccelerationBall : MonoBehaviour

{

private Ball \_ball;

public float cofficient = 1.5f;

public int accelerationTime = 5;

private void Start()

{

gameObject.GetComponent<EventGame>().onEvent += Play;

}

private void OnDestroy()

{

gameObject.GetComponent<EventGame>().onEvent -= Play;

}

public void Play()

{

Debug.Log("AccelerationMove");

\_ball = BallController.Instance.GetActiveBall().GetComponent<Ball>();

StartCoroutine(AccelerationMove());

}

IEnumerator AccelerationMove()

{

float timer = 0f;

\_ball.Move(false, cofficient);

while (timer < accelerationTime)

{

yield return null;

timer += Time.unscaledDeltaTime;

}

\_ball.Move();

}

}

================

using System;

using UnityEngine;

public class PlatformExtension : MonoBehaviour

{

private Platform \_platform;

private void Start()

{

gameObject.GetComponent<EventGame>().onEvent += Play;

}

private void OnDestroy()

{

gameObject.GetComponent<EventGame>().onEvent -= Play;

}

public void Play()

{

Debug.Log("PlatformExtension");

\_platform = GameController.Instance.platform;

var scale = \_platform.transform.localScale;

if(scale.x.Equals(1))

{

scale = new Vector3(2, scale.y, scale.z);

\_platform.leftBorder += 1f;

\_platform.rightBorder -= 1f;

}

else

{

scale = new Vector3(1, scale.y, scale.z);

\_platform.leftBorder = -1.7f;

\_platform.rightBorder = 1.7f;

}

\_platform.transform.localScale = scale;

}

# **ПРИЛОЖЕНИЕ Б**

**Организационная структура организации**

