КОМИТЕТ ПО ОБРАЗОВАНИЮ МИНГОРИСПОЛКОМА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

МИНСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ ЭЛЕКТРОНИКИ

Специальность: 2-40 01 01

"Программное обеспечение информационных технологий"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

РАЗРАБОТКА ПРИЛОЖЕНИЯ «ZHGI!»

КП38ТП.011102.081ПЗ

Автор проекта (О.Б.Жгуновский)

Руководитель проекта (Ж.В.Леус )

2021

Учреждение образования

«Минский государственный колледж электроники»

УТВЕРЖДАЮ

Председатель цикловой (предметной)

комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« » \_\_\_\_\_\_\_\_\_ 20 \_\_\_г.

**ЗАДАНИЕ**

**на курсовой проект (работу)**

Учащейся *Жгуновскому Олегу Борисовичу*

(фамилия, имя отчество)

курса 3 группы 38

по учебной дисциплине «Конструирование программ и языки программирования»

Тема курсового проекта (работы)*Разработка приложения «ZHGI!»*

Исходные данные *сведения об игровом процессе*

*Техническое задание: Создать приложение, основой которого является приложение типа «кликер», имеющее возможность бесконечно улучшать игровой процесс. Приложение должно иметь эргономичный графический интерфейс и корректно реагировать на запросы пользователя.*

**Состав проекта (работы)**

**Пояснительная записка**

|  |  |
| --- | --- |
| Содержание разделов | Срок выполнения |
| *Анализ задачи* | *до 30.03* |
| *Проектирование задачи* | *до 10.04* |
| *Реализация* | *до 20.04* |
| *Применение* | *до 30.04* |
| *Список использованных источников* | *до 10.05* |

**Графическая часть проекта**

Лист -1 *Диаграмма* *классов*

Лист -2 *Диаграмма вариантов использования*

Дата выдачи « 10 » марта 20 21 г.

Срок сдачи « 15 » мая 20 21 г.

**Преподаватель-руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Ж.В.Леус**

**курсового проекта**

**Подпись учащегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /О.Б.Жгуновский**

СОДЕРЖАНИЕ

[Введение 4](#_Toc67130453)

[1 Анализ задачи 5](#_Toc67130454)

[1.1 Описание предметной области 5](#_Toc67130455)

[1.2 Функциональная модель 5](#_Toc67130456)

[1.3 Модель данных 6](#_Toc67130457)

[1.4 Инструменты разработки 7](#_Toc67130458)

[1.5 Требования к программе 7](#_Toc67130459)

[1.6 Системные требования 8](#_Toc67130460)

[2 Проектирование задачи 9](#_Toc67130461)

[2.1 Организация данных 9](#_Toc67130462)

[2.2 Процессы 11](#_Toc67130463)

[2.3 Концептуальный прототип 11](#_Toc67130464)

[2.4 Система помощи 11](#_Toc67130465)

[2.5 Тесты на использование 11](#_Toc67130466)

[3 Реализация 13](#_Toc67130467)

[3.1 Функции: логическая и физическая организация 13](#_Toc67130468)

[3.2 Функция и элементы управления 13](#_Toc67130469)

[3.3 Функциональное тестирование 13](#_Toc67130470)

[3.4 Полное тестирование 14](#_Toc67130471)

[4 Применение 16](#_Toc67130472)

[4.1 Назначение 16](#_Toc67130473)

[4.2 Описание интерфейса программы 16](#_Toc67130474)

[4.3 Среда функционирования 16](#_Toc67130475)

[4.4 Инсталляция 18](#_Toc67130476)

[Заключение 19](#_Toc67130477)

[Список использованных источников 20](#_Toc67130478)

[Приложение А 21](#_Toc67130479)

[Приложение Б 24](#_Toc67130480)

[Приложение Г 25](#_Toc67130481)

# ВВЕДЕНИЕ

Увеличение доходов от мобильных игр продолжит опережать рост доходов от продажи ПК игр в следующем году, что в конечном итоге приведет к сокращению доли рынка игр для ПК до 20% в 2022 году. Кроме этого, доход от мобильных игр также опережает доход от игр на консолях.

В 2020 году Newzoo прогнозирует, что на игры для ПК будет приходиться не более 20% дохода рынка, для консольных игр – не более 30%, а для мобильных устройств и планшетов от 46%. А к 2022 году на мобильные и планшетные игры будет приходиться около 50% всего игрового рынка.

Происходит это по разным причинам, но в основном, из-за того, что мобильная платформа более доступна нежели остальные, а также легче в освоении.

Цель игр на мобильном устройстве – позволить пользователю скоротать время.

Цель моей разработки «ZHGI!» предназначена для развлечения пользователя, в особо скучные минуты его жизни.

Данному программному средству присущи следующие свойства:

– простота освоения;

– удобное представление информации;

– эргономичный интерфейс.

Данная пояснительная записка является полным описанием разработанного программного продукта. Она разделена на разделы и подразделы, и каждый раздел или подраздел посвящен определённому аспекту процесса разработки приложения.

В первом разделе описывается анализ задач.

Во втором разделе описывается проектирование задачи.

В третьем разделе описывается реализация задачи.

В четвертом разделе описывается применение программы.

# Анализ задачи

# Описание предметной области

Техническое задание данной работы требует разработать программу на языке программирования С#.

Программное обеспечение предназначена для развлечения пользователя. Данная программа должна иметь простой и удобный интерфейс.

Программа должна выглядеть эргономично, иметь дружественный интерфейс, любые действия должны быть интуитивно понятными и не вызывать трудностей при работе с программой.

# Функциональная модель

Данный программный продукт предоставляет следующие основные функции:

– настройки программы:

а) регулирование громкости музыкального сопровождения;

б) регулирование громкости звуковых эффектов;

– игровой процесс:

а) посадка дерева;

б) получение внутриигровой валюты за продажу выращенного дерева;

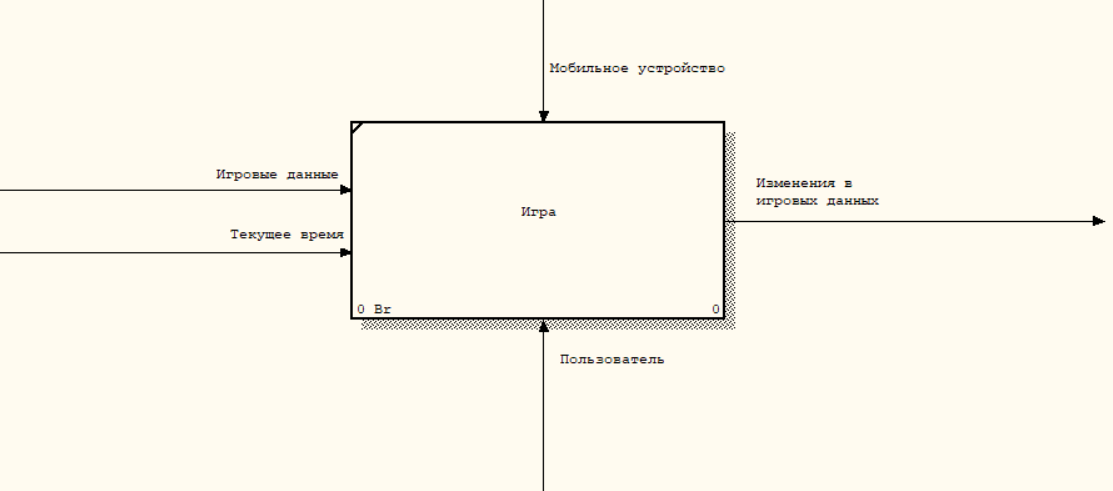
– система прогресса:

а) сохранение прогресса текущей игровой сессии;

б) покупка улучшений для ускорения прогресса;

в) выбор улучшений из трёх предлагаемых категорий.

Функциональная модель представлена на рисунке 1.1.

Рисунок 1.1 – Функциональная модель

# Модель данных

Модель данных данной программы строится на основе взаимодействия графической оболочки, базы данных и пользователя. В первую очередь пользователю необходимо выбрать желаемое дерево для посадки.

После этого, запустится таймер, который будет отсчитывать время до того момента, пока внутриигровое дерево не вырастет.

Когда дерево вырастет, программа «продаст» его, а пользователю на баланс будет зачислена внутриигровая валюта, в зависимости от типа дерева и улучшений пользователя.

Для ускорения процесса получения валюты, пользователь может купить улучшения, которые могут уменьшить стоимость семян, увеличить доход от продажи дерева или ускорить время его роста.

Наиболее часто при разработках баз данных применяются реляционные СУБД. Для СУБД этого типа модель удобно представить в виде схемы данных на рисунке 1.2.

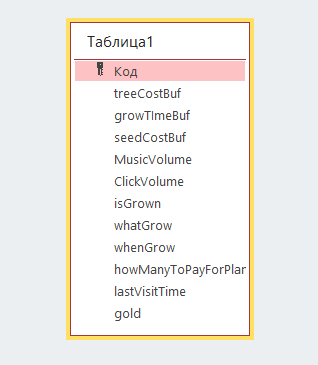


Рисунок 1.2 – Схема данных

# Инструменты разработки

Среда разработки: Microsoft Visual Studio 2019, Unity.

Язык программирования: C#.

Для реализации поставленной задачи в качестве среды разработки была выбрана Unity. Данный движок предоставляет визуальную среду разработки, поддержку межплатформенной и модульность своей системы.

Для реализации, поставленной задачи в качестве среды программирования для разработки приложения была выбрана Microsoft Visual Studio 2019. Проектирование в данной среде является очень удобным, т.к. обладает широким набором возможностей: создание форм, работа с изображениями, создание кнопок, огромный функционал для работы с базами данных и т.д.

Фрагмент текста программы представлен в приложении А.

# Требования к программе

Программа должна быть разработана на языке программирования C#.

Функциональные требования:

– реагирование на нажатие пользователя;

– покупка улучшений;

– ускорение прогресса за счёт покупки улучшений.

Под улучшениями подразумеваются внутриигровые покупки, которые можно совершать за внутриигровую валюту.

Программа должна иметь интуитивно понятный графический интерфейс и не вызывать трудностей при работе, корректно реагировать на действия пользователя.

# Системные требования

Рекомендуемые системные требования:

– Android устройство с версией прошивки не ниже “Android 5.0 Lollipop”.

# Проектирование задачи

# Организация данных

Данные сохраняются непосредственно при работе с приложением. Ввод данных осуществляется с помощью диалоговых окон. Данные сохраняются при каждом из изменений.

Организация интерфейса – это очень важное свойство приложения. Интерфейс должен обладать следующими свойствами: гибкостью, простотой, естественностью.

Основной интерфейс приложения будут представлять окна. Все окна кроме главного, будут содержать пользовательские элементы управления. Работа приложения будет начинаться с главного окна, где пользователь может начать игру.

При создании приложения в Unity важную роль играют сцены, так как только они являются основным диалоговым средством работы пользователя. Сцены содержат объекты игры. Они могут использоваться для создания главного меню, отдельных уровней и для других целей. Можно считать каждый файл сцены отдельным игровым уровнем. В каждой сцене можно разместить объекты окружения, декорации, по кусочкам создавая дизайн и саму игру.

Разрабатываемое приложение имеет одну основную сцену.

Особое внимание нужно обратить на главное меню, так как оно является главным окном, оно должно позволять легко переходить на другие окна.

Данные пользователей и их настройки хранятся в сокрытом виде в регистре.

Пример процесса сохранения настроек пользователя в регистр, представлен в соответствии с рисунком 2.1.



Рисунок 2.1 – Пример сохранения настроек

База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица.

Таблица хранит информацию о настройках пользователя, его прогрессе и таймер. Структура данных приведена в таблице 2.1. Вся информация хранится в зашифрованном виде.

Таблица 2.1 — Структура таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип поля | Размерность | Наименование  (смысл отображаемой информации) | Примечание |
| treeCostBuf | string | 256 | Строка, в которой содержатся все улучшения, купленные пользователем из категории «стоимость продажи дерева» |  |
| growTimeBuf | string | 256 | Строка, в которой содержатся все улучшения, купленные пользователем из категории «время роста дерева» |  |
| seedCostBuf | string | 256 | Строка, в которой содержатся все улучшения, купленные пользователем из категории «стоимость покупки дерева» |  |
| MusicVolume | float | 1 | Число, в котором содержится громкость музыкального сопровождения |  |
| ClickVolume | float | 1 | Число, в котором содержится громкость звуковых эффектов |  |
| isGrown | string | 256 | Строка, в которой содержится информация о том, растёт ли дерево |  |
| whatGrow | string | 256 | Строка, в которой содержится информация о том, какое дерево растёт |  |
| whenGrow | string | 256 | Строка, в которой содержится информация о том, когда вырастет дерево |  |
| howManyToPayForPlant | string | 256 | Строка, в которой содержится информация о том, сколько платить за каждое из деревьев |  |
| lastVisitTime | string | 256 | Строка, которая содержит время последнего визита |  |
| gold | int | 32 | Строка, которая содержит баланс пользователя |  |

# Процессы

В разработке данного программного продукта можно выделить следующие процессы:

– проектирование структуры приложения;

– планирование интерфейса взаимодействия пользователя с приложением;

– обработка исключительных ситуаций;

– тестирование функций.

# Концептуальный прототип

Прототипирование представляет собой описание внешнего пользовательского интерфейса – систему меню, кнопок, диалогов, окон.

При создании приложения важную роль играют окна, только они являются основным диалоговым средством работы пользователя. Меню содержит следующие пункты:

– «Посадить дерево» – нажав на кнопку пользователь сможет перейти к выбору деревьев, и выбрать что садить;

– «Настройки» – нажав на кнопку пользователь получит возможность изменить громкость музыкального сопровождения;

– «Купить улучшение» – нажав на кнопку пользователь потратит свою внутриигровую валюту и приобретёт улучшение, ускоряющее его прогресс в игре;

– «Изменить меню» – нажав на одну из кнопок, пользователь изменит текущий список улучшений, на другой.

# Система помощи

Система помощи отсутствует в связи с тем, что предполагается, что пользователь умеет нажимать на экран и без сторонней помощи.

# Тесты на использование

Для тестирования данного программного продукта можно предложить следующие тесты:

* тестирование запуска и закрытия программы;
* тестирование изменения громкости музыкального сопровождения;
* тестирование сохранения прогресса;
* тестирование корректного поведения графического интерфейса;
* тестирование корректного приобретения улучшений;
* тестирование корректного приобретения семян для посадки;
* тестирование корректной загрузки приложения.

# Реализация

# Функции: логическая и физическая организация

Логическая функциональная модель включает в себя прежде всего функции по добавлению и удалению контента, с возможностью последующего обращения к этим данных. Кроме того, реализованы функции, не требующие сохранения результата, такие, как конвертеры, калькулятор, обозреватель часовых поясов.

Физическая реализация функций, требующих сохранения результата, сводится к разработке диалога с пользователем. Реализация функций, не требующих сохранения результата, сводится к исключительно математическим расчетам.

# Функция и элементы управления

При создании использовались следующие элементы:

– «Text» – элемент для вывода текста на экран в виде надписи;

– «Button» – элемент, нажатие на который вызывает действие;

– «Image» – контейнер, в который можно поместить любое графическое изображение;

– «Panel» – графический элемент в виде прямоугольника;

– «Slider» – элемент, позволяющий динамически регулировать некоторое значение в определённом диапазоне;

– «Audio source» – абстрактный элемент, позволяющий проигрывать звук;

– «Canvas» – представляет собой абстрактное пространство, в котором производится настройка и отрисовка UI;

– «Camera» – пространство, в котором отрисовывается всё что видит пользователь;

– «Scrollbar» – элемент, который при добавлении его в контейнер позволяет прокручивать содержимое данного контейнера.

# Функциональное тестирование

В ходе функционального тестирования на этапе разработке были выявлены недостатки и отсутствие некоторых функционально важных элементов.

В ходе тестирования были обнаружены программные ошибки и ошибки в интерфейсе. Функциональное тестирование выявило слабые места в программе, что помогло устранить и доработать программу. Устранение и доработка функциональности программы была произведена в рамках технического задания, с добавлением дополнительного функционала, не противоречащего требованиям.

После внесения доработок и поправок программа прошла повторное тестирование, не выявившее недочетов на данном этапе.

Из выше сказанного можно сделать вывод, что программа полностью функционирует и пригодна для решения задач.

# Полное тестирование

Разработка любой программы предполагает наличие ошибок в исходном тексте и борьбу с ними. Ошибки обычно подразделяют на три группы:

* синтаксические ошибки;
* ошибки времени выполнения программы;
* смысловые (логические) ошибки.

Результаты проведенного тестирования приведены в таблице 3.1.

Таблица 3.1 – Результаты тестирования программного продукта

|  |  |  |
| --- | --- | --- |
| Название теста | Последовательность действий | Результат |
| Проверка установки программы | Открытие файла «ZHGI.apk» | Программа установлена |
| Запуск | Нажатие на иконку ZHGI | Запуск программы |
| Проверка работоспособности кнопок | Перемещение по разделам программы | Все разделы функционируют |
| Проверка изменения громкости | Открытие меню «Настройки», перемещение ползунка | Громкость изменена |
| Проверка повторного входа | Запуск иконки clicker | Прогресс успешно восстановлен |
| Проверка покупки улучшения «Ускорение роста» | В коричневом нижнем меню покупка улучшения. Посадка дерева | Время роста было уменьшено |

В процессе тестирования проверялась правильность функционирование программы.

Тестирование проекта позволяет сделать вывод, что программа удовлетворяет требованиям, предъявляемым к интерфейсу взаимодействия с пользователем.

При полном тестировании моделируются все возможные действия пользователя при работе с программой. В полном тестировании должен быть описан тестовый пример с конкретными входными данными и результатами, которые должны быть получены в результате работы приложения.

# Применение

# Назначение

Программа для развлечения пользователя.

# Описание интерфейса программы

Система игры построена таким образом, что при открытии программы clicker на рабочем столе телефона открывается стартовое окно, которое представлено на рисунке 4.1, где можно начинать играть.



Рисунок 4.1 – Основное меню приложения

При выборе иконки «настройки» откроется следующее окно как показано на рисунке 3.2.

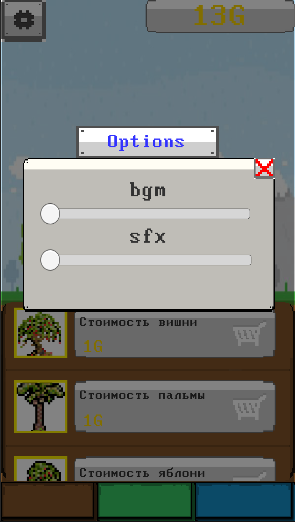


Рисунок 4.2 – Окно раздела «настройки»

Для окончания настройки надо нажать кнопку «Закрыть». После закрытия, пользователь возвращается на главное окно.

Для покупки дерева, необходимо нажать на знак плюса, и выбрать дерево из открывшегося меню. Меню для выбора дерева представлена на рисунке 4.3.

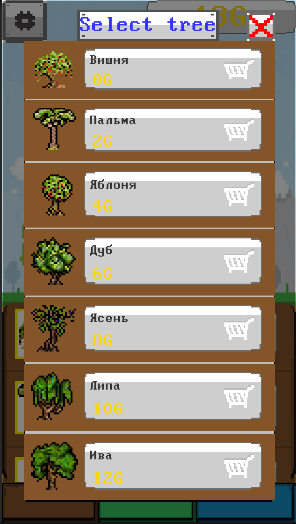


Рисунок 4.3 – Окно выбора дерева.

После посадки, вместо плюса появляется изображение дерева, и над ним таймер с отображением времени через которое оно вырастет. Пример показан на рисунке 4.4.

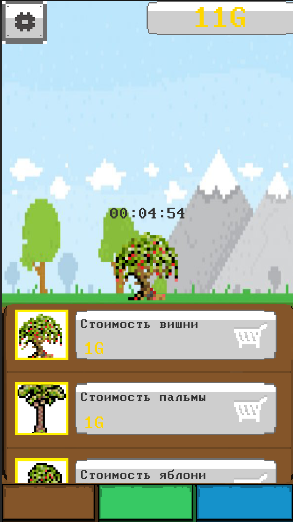


Рисунок 4.4 – После посадки дерева

Когда дерево высадит, пользователю зачислится внутриигровая валюта. С её помощью он может купить одно из улучшений, представленных в трёх секциях нижнего меню, или посадить более дорогое дерево.

# Среда функционирования

Средой функционирования является операционная система Android.

# Инсталляция

Для полноценного использования данного программного продукта, необходимо переписать apk архив на память андроид-устройства, и запустить файл «ZHGI.apk». После чего произвести инсталляцию архива.

# ЗАКЛЮЧЕНИЕ

В ходе написания данной программы были получены знания работы с языком C# и игровым движком Unity. Кроме того, были выявлены и устранены ошибки и недостатки программы. Проведено функциональное и полное тестирование, в ходе которых программа приобрела законченный вид полностью работающего продукта.

Данный курсовой проект создан для развлечения пользователя. Пользоваться программой может любой человек с минимальными навыками обращения с мобильным устройством.

Программа является полноценным программным продуктом и может быть доработана и усовершенствована дополнительными функциональными возможностями или расширением области изучения команд и языков.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по выполнению проектов Специальность: 2-40 01 01 «Программное обеспечение информационных технологий».
2. Троелсен, Эндрю. Язык программирования C# 2010 и платформа .NET 4, Изд. APress, 2017 г. — 1392 c.
3. Албахари, Джозеф, Албахари, Бен – C# 6.0. Справочник. Полное описание языка – Вильямс, 2015 – 1040 с.
4. Зиборов, В. В. – Visual C# 2012 на примерах – БХВ-Петербург, 2017 – 475 с.
5. Дейтел, Пол, Дейтел, Харви – Как программировать на Visual C# 2012 – Спб.:Питер, 2018 – 858 с.
6. MSDN MicroSoft [Электронный ресурс]. – Электронные данные. – Режим доступа: http://msdn.microsoft.com.
7. Unity Learn [Электронный ресурс] – Электронные данные. – Режим доступа: https://learn.unity.com/
8. C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. — 1056 с.: ил. — Парал. тит. англ.

# ПРИЛОЖЕНИЕ А

(обязательное)

Текст программы

Модуль «Updates.cs»

using System;

using System.Linq;

using UnityEngine;

using UnityEngine.UI;

public class Upgrades : MonoBehaviour

{

//будем хранить левела, а потом пихать на |ln(x)|. Стоимость будет |ln(x\*2)|

public static int[] treeCostBuf = new int[] { 1, 1, 1, 1, 1, 1, 1 };//стоимость продажи дерева

public static int[] growTimeBuf = new int[] { 1, 1, 1, 1, 1, 1, 1 };//время роста вишни

public static int[] seedCostBuf = new int[] { 1, 1, 1, 1, 1, 1, 1 };//стоимость семечка

public Button[] senders0;//нулевое меню

public Button[] senders1;//первое меню

public Button[] senders2;//второе меню

public Text[] cost0;

public Text[] cost1;

public Text[] cost2;

public void BuyUpgrade(Button sender)

{

Upgrade(sender);

ShowPrice();

Save();

}

private void ShowPrice()//отображает текущую цену бафов

{

for (int i = 0; i < treeCostBuf.Length; i++)

{

cost0[i].text = Convert.ToInt32(Math.Abs(Math.Log(treeCostBuf[i] \* 2, Math.E))).ToString() + "G";

cost1[i].text = Convert.ToInt32(Math.Abs(Math.Log(growTimeBuf[i] \* 2, Math.E))).ToString() + "G";

cost2[i].text = Convert.ToInt32(Math.Abs(Math.Log(seedCostBuf[i] \* 2, Math.E))).ToString() + "G";

}

}

private void Upgrade(Button sender)//служит для улучшения бафа

{

//теперь вроде должно работать только если голды положительная сумма

for (int i = 0; i < treeCostBuf.Length; i++)

{

if (sender == senders0[i])

{

if (GameProgress.gold - Math.Abs(Math.Log(treeCostBuf[i] \* 2, Math.E)) >= 0) // проверяем чтобы денег хватило через | ln(x \* 2) |

{

treeCostBuf[i]++;

GameProgress.gold -= Convert.ToInt32(Math.Abs(Math.Log(treeCostBuf[i] \* 2, Math.E)));

}

return;

}

}

for (int i = 0; i < growTimeBuf.Length; i++)

{

if (sender == senders1[i])

{

if (GameProgress.gold - Math.Abs(Math.Log(growTimeBuf[i] \* 2, Math.E)) >= 0) // проверяем чтобы денег хватило через | ln(x \* 2) |

{

growTimeBuf[i]++;

GameProgress.gold -= Convert.ToInt32(Math.Abs(Math.Log(growTimeBuf[i] \* 2, Math.E)));

}

return;

}

}

for (int i = 0; i < seedCostBuf.Length; i++)

{

if (sender == senders2[i])

{

if (GameProgress.gold - Math.Abs(Math.Log(seedCostBuf[i] \* 2, Math.E)) >= 0) // проверяем чтобы денег хватило через | ln(x \* 2) |

{

seedCostBuf[i]++;

GameProgress.gold -= Convert.ToInt32(Math.Abs(Math.Log(seedCostBuf[i] \* 2, Math.E)));

}

return;

}

}

}

private void Start()

{

Load();

ShowPrice();

}

private void Load()

{

string txt = PlayerPrefs.GetString("treeCostBuf");

if (txt != "")

treeCostBuf = GetIntArrFromFormatString(txt);

txt = PlayerPrefs.GetString("growTimeBuf");

if (txt != "")

growTimeBuf = GetIntArrFromFormatString(txt);

txt = PlayerPrefs.GetString("seedCostBuf");

if (txt != "")

seedCostBuf = GetIntArrFromFormatString(txt);

}

private int[] GetIntArrFromFormatString(string txt)//получаем массив из строки после загрузки

{

return txt.Split(' ').Select(Int32.Parse).ToArray();

}

private void Save()

{

string txt = GetFormatStringFromArr(treeCostBuf);

PlayerPrefs.SetString("treeCostBuf", txt);

txt = GetFormatStringFromArr(growTimeBuf);

PlayerPrefs.SetString("growTimeBuf", txt);

txt = GetFormatStringFromArr(seedCostBuf);

PlayerPrefs.SetString("seedCostBuf", txt);

PlayerPrefs.Save();

}

private string GetFormatStringFromArr(int[] arr)//возвращаем форматированую строку из массива интов для сейва

{

string txt = null;

for (int i = 0; i < arr.Length; i++)

{

txt += arr[i] + " ";

}

txt = txt.Remove(txt.Length - 1);//прочекать

return txt;

}

}

# ПРИЛОЖЕНИЕ Б

(справочное)

Диаграммы UML

(диаграмма вариантов использования)



# ПРИЛОЖЕНИЕ Г

(справочное)

Диаграммы UML

(диаграмма классов)

