МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Северо-Восточный государственный университет

ИНСТИТУТ цифровых технологий и экономики

кафедра математики и информатики

Зарегистрировано:

№ \_\_\_\_\_\_\_\_/ к/р

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_20\_\_\_\_г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

#### Курсовая работа

по дисциплине Б1.Б.21 «Программирование на языках высокого уровня»

**Разработка приложения «Гор\_Город»**

Выполнил:

студент 3 курса гр. ПИБ-71

Казаков Кирилл Сергеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

Руководитель:

Старший преподаватель кафедры информатики СВГУ

Марсенич И.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, подпись)

Курсовая работа представлена «\_\_\_\_» \_\_\_\_\_\_ 2020 г.

Магадан

2020

## ОГЛАВЛЕНИЕ

[**ВВЕДЕНИЕ** 3](#_Toc31154823)

[**1.** **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ** 5](#_Toc31154824)

[1.1. Принципы разработки ПО 5](#_Toc31154825)

[1.2. Анализ существующих аналогов разрабатываемого продукта 6](#_Toc31154826)

[1.3. Необходимые сведения из теории сельского хозяйства 7](#_Toc31154827)

[1.4. Постановка задачи 7](#_Toc31154828)

[**2.** **РАЗРАБОТКА ПРИЛОЖЕНИЯ «Гор\_город»** 10](#_Toc31154829)

[2.1 Описание работы алгоритма программы 10](#_Toc31154830)

[2.2 Графический интерфейс программы 10](#_Toc31154831)

[2.3 Программный код 12](#_Toc31154832)

[2.4 Описание работы программы 20](#_Toc31154833)

[**ЗАКЛЮЧЕНИЕ** 24](#_Toc31154834)

[**БИБЛИОГРАФИЯ** 25](#_Toc31154835)

# **ВВЕДЕНИЕ**

На данный момент времени вычислительная техника играет все большую роль в жизни современного общества. Практически любая сфера деятельности, так или иначе, подразумевает в себе различное взаимодействие с информационными технологиями [4], осуществляются такие процессы как: сбор, хранение, обработка, передача и использование данных. Потребность в использовании информационных технологий возникает из-за того, что они позволяет намного удобнее, проще и надежнее осуществлять вышеперечисленные процессы. Они необходимы для результативной организации рабочего процесса, которая способствует уменьшению затрат времени, снижению количества действий и всяческих ценных ресурсов.

Для внедрения информационных технологий (ИТ) в различные сферы деятельности разрабатывается ИТ-специалистами и используется разностороннее прикладное программное обеспечение (ПО), которое обладает большим функционалом и выполняет те или иные поставленные перед ним задачи. Прикладное ПО может выполнять как универсальные, так и узконаправленные задачи.

На данный момент создается все больше прикладного ПО для осуществления различных задач, однако уже достаточно продолжительный период времени в сельскохозяйственной отрасли существует проблема, связанная с отсутствием такого ПО, которое бы позволяло осуществлять изменение, хранение и обработку данных над необходимыми объектами. На настоящий момент для обработки и хранения небольшого количества данных в электронных таблицах (относительно различных систем управления базами данных(СУБД)) существуют такие программы, как Microsoft Excel; для операций с крупным объемом данных существует большое количество различных СУБД, таких как MySQL.

Однако и СУБД, и электронные таблицы обладают достаточно большим функционалом, без изучения возможностей которого использование таких программных средств затруднительно, при этом, большая часть данного ПО остается незадействованной при решении каких-либо конкретных задач. Вследствие этого, человек, не имеющий опыта работа с данным ПО, должен потратить некоторое количество времени на его изучение или же нанять ИТ-специалиста, разбирающегося в этом ПО.

Целью курсовой работы является создание приложения «Гор\_город» с графическим интерфейсом, которое будет осуществлять учет посадок земельного участка.

Объектом исследования является технология программирования Windows-приложений в межплатформенной среде разработки компьютерных игр Unity с использованием объектно-ориентированного языка программирования C#.

Предметом исследования является разработка приложения «Гор\_город» для внесения описания данных земельного участка и их дальнейшее хранение.

Задачи курсовой работы:

* рассмотреть основные принципы программирования;
* проанализировать критерии качества программы;
* провести анализ аналогичных приложений;
* изучить предметную область, связанную с рациональным использованием земельного участка;
* рассмотреть общие принципы разработки на платформе Unity, используя язык C#;
* научиться связывать компоненты визуальной среды с программным кодом;
* разработать Windows приложение «Гор\_город»;
* провести отладку и тестирование созданного приложения.

Курсовая работа состоит из введения, двух глав, заключения, списка источников и приложения на CD-диске.

# **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

## Принципы разработки ПО

Разработка любого приложения основывается на некоторых базовых принципах [1], [2]. Порядок разработки приложения происходит согласно его размеру и свойствам, а также определяется тем, чего от него требует клиент. Действия будут различаться, возможна разработка приложений на разные платформы, разработка встроенного ПО, а также решения для автоматизации Баз Данных (БД), но выполняемые в процессе разработки этапы всегда будут примерно одинаковы. [3]

**Этап подготовки.** На данном этапе решаются следующие вопросы:

* Какую информацию может предоставить заказчик, в какой степени предоставленная им информация достаточна, составляется вывод об объеме будущей работы. Далее делаются макеты того, как именно будет выглядеть ПО (интерфейс, схема работы, структура БД и т. д).
* Контроль: архитектор по возможности выполняет все требования менеджера проектов.
* Утверждение: клиент удостоверяется в том, что заданные им условия совпадают с задекларированными исполнителем возможностями приложения.

**Проектирование.** Результатом данного этапа является техническое задание (ТЗ) с четко сформулированной для заказчика и исполнителя (группы людей, работающих над процессом разработки) иллюстрацией ответов на вопросы:

* Какую работу нужно проделать для выполнения заданной цели (описание функционала продукта и пользователей, для которых он предназначен).
* Какой должна быть структура будущего ПО.
* Как проверить достижение заданной перед разработчиками задачи (верификация, критерии оценки).

**Этап создания** делится на следующие этапы:

Дизайн – наряду с техническими возможностями программного обеспечения дизайн является важнейшей составляющей, которая также влияет на качество взаимодействия пользователя с ним. Дизайн должен создаваться согласно таким первостепенным требованиям как простота, интуитивность, соответствие заданному заказчиком стилю. Также дизайн необходимо выполнить таким образом, чтобы для достижения конечного результата пользователь осуществлял бы минимально возможное количество затрат.

Код — данный этап, как правило, сопоставляется с [разработкой ПО](http://www.edsd.ru/ru/proekty/razrabotka_po). Необходимо чтобы код был оптимизированным, удобным и понятным. Осуществляется выбор такого языка программирования, работать с которым будет максимально удобно и, одновременно с этим, данный язык будет иметь достаточный спектр возможностей, который позволил бы выполнить поставленную перед исполнителем задачу.

[**Тестирование**](http://www.edsd.ru/ru/uslugi/testirovanie_po). Следующий этап должен осуществляться на каждом участке разработки ПО. Он включает в себя множество различных тестов по [плану верификации](http://www.edsd.ru/public/files/%D0%9F%D0%BB%D0%B0%D0%BD_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B.pdf), выполняющимся с учётом специфики проекта на этапе составления ТЗ.

[**Документирование**](http://www.edsd.ru/ru/princypy/documentirovanie) — это процедура, необходимая для фиксации плана, процесса и результата разработки ПО. Этот этап включает всю начальную информацию (ТЗ, макеты), планы работ, затрат, тестирования, задачи исполнителей в каждый момент времени и так далее. Документация нужна, чтобы быстро и точно находить ошибки в разработанном ПО.

**Этап поддержки** делится на следующие подэтапы:

* Внедрение – настройка ПО под заданные условия использования, а также ознакомления пользователей с функционалом программного продукта.
* Сопровождение ПО – это улучшение, оптимизация и избавление от недостатков программного обеспечения после начала его эксплуатирования.

## Анализ существующих аналогов разрабатываемого продукта

Был проведен анализ приложений, реализующих аналогичный функционал. С похожим функционалом было найдено одно приложение Garden Time. Это приложение имеет гораздо больший функционал, однако имеет ряд существенных недостатков:

* Отсутствие локализации (в наличие только версия на английском языке.
* Строгие вводимые параметры (наличие только определенного списка разрабатываемых культур).
* Отсутствие графического отображения полей, на которых будут разрабатываться культуры. Конечный пользователь не сможет визуально изобразить поля, вследствие этого ему будет сложнее запомнить местоположения посаженных культур.
* Из-за наличия чрезмерно большого функционала использование данного программного продукта может показаться пользователю затруднительным. Тем не менее стоит признать, что изучить и использовать такое ПО будет гораздо легче, чем обычные электронные таблицы, так как в приложении присутствует интуитивно-понятный графический интерфейс, разрабатываемый специально для пользователей, работающих в сельскохозяйственной отрасли.

## Необходимые сведения из теории сельского хозяйства

Рассматриваемой предметной области является отрасль малого сельского хозяйства. В дальнейшем под пользователями ПО будут подразумеваться люди, осуществляющие взаимодействие с земельным участком. [5]

Земельный участок (согласно Земельному кодексу России) – это участок, предоставленный гражданину или приобретенный им для выращивания ягодных, овощных бахчевых или других сельскохозяйственных культур и картофеля. Земельный участок разделен на поля, на которых, в свою очередь, возделываются различные посевные культуры, при этом одинаковые культуры могут иметь разный сорт.

При всем разнообразии проблем в данной сфере одной из основных является журнализация характеристик полей земельного участка в конце сезона работ. При отсутствии журнализации, люди, осуществляющие работу на участке, не смогут точно запомнить месторасположение той или иной культуры на одном из полей, при этом желательно также запомнить сорт и характеристики данной культуры, такие как урожайность, количество потраченных удобрений и количество обработки растений от паразитов и заболеваний.

Урожайность – это сумма собранных с определенного поля единиц культуры.

Количество потраченных удобрений - это количество использованных минералов, необходимое для сбора такого урожая, который будет удовлетворять работника. При этом чем меньше удобрений было затрачено для возделывания какой-либо культуры, тем лучше, так как для достижения наиболее качественного продукта на выходе, культуры должны расти естественным образом.

Количество обработки растений от паразитов и заболеваний – это такое количество использованных антивредоносных средств, которое необходимо для выживания посаженных культур. Также, как и в случае с удобрениями, чем меньших таких средств было потрачено, тем лучше результат (продукт на выходе).

При чрезмерном использовании как удобрений, так и антивредоносных средств для обработки посевной культуры рекомендуется либо отказаться от возделывания данной культуры, либо изменить условия взращивания.

## Постановка задачи

Для журнализации полей аграрного участка предназначено приложение «Гор\_город». Приложение должно создавать визуальную модель земельного участка, делящегося на поля. Пользователю предоставлена возможность изменять параметры любого из полей, а также сохранять введенные им значения, то есть пользователь может осуществлять учет данных о полях, составить общую статистику за год.

**Спецификация входных и выходных данных:** на вход в программу поступает несколько значений, вводимых пользователем. В первую очередь поступает число, обозначающее количество полей в участке. Далее пользователь присваивает каждому из полей все необходимые ему параметры, а именно: наименование культуры, сорт, урожайность, количество потраченных удобрений и количество обработки растений от паразитов и заболеваний.

На выходе пользователь имеет все введенные им данные для каждого поля. Также все введенные данные сохраняются при выходе из приложения.

**Выбор средств программирования:** для того, чтобы создать данное приложение, было необходимо выбрать такое ПО, в котором было бы наиболее удобно связывать визуально-понятный графический интерфейс и выполняемый им функционал. В результате размышлений по этому поводу мною была выбрана платформа Unity, предназначенная для разработки компьютерных и мобильных игр, а следствие, в ней можно максимально просто и удобно связывать различные визуальные компоненты с написанным для них кодом. В качестве языка программирования мною был выбран язык C#, так как:

1) Данный язык является объектно-ориентированным [16], [17]. Это позволяет создавать в программе классы (шаблоны, по которым будет описываться отдельные экземпляры в случае их создания. Экземпляры класса, имеют поля, в которые записываются или с которых считываются значения определенного типа. Также экземпляр класса может выполнять определенные методы, заложенные в него при описании класса. Конкретно в случае данной курсовой работы, каждое поле участка будет являться экземпляром класса, который, в свою очередь, будет иметь различные характеристики различного типа (наименование культур, сорт, урожайность и т.д.).

2) В программах, написанных с применением объектно-ориентированного подхода удобнее производить рефакторинг, то есть изменять программный код с целью облегчения понимания его работы без написания новой функциональности, так как все объекты – это отдельные элементы, содержащие в себе собственные свойства и методы.

3) Программы, написанные с применением ООП, позволяют абстрагироваться не только при написании каких-либо функций, но и представить объект в качестве набора абстрактных характеристик, которые будут инициализироваться непосредственно при создании экземпляра. При этом поведение объекта будет описываться методами класса, которые также могут быть вызваны после создания экземпляра класса [9].

4) Так как C# является на сегодняшний день одним из самых популярных языков программирования, то при возникновении какой-либо проблемы не составит труда найти практически любой материал по интересующей теме [7], [20].

В конечном итоге, для создания программы «Гор\_город» понадобился компьютер на операционной системе Windows, среда разработки Microsoft Visual Studio [13] и платформа Unity версии 5.1 [6].

# **РАЗРАБОТКА ПРИЛОЖЕНИЯ «Гор\_город»**

## Описание работы алгоритма программы

Алгоритм работы приложения осуществляется следующим образом:

1. Проверяется наличие сохранения результата работы, если оно имеется, то пользователю выводятся его уже заполненные поля (переход к этапу 4), если нет, то:
2. На вход подается некоторое число k, обозначающее количество грядок, которые должны появиться.
3. При нажатии на кнопку «Создать» генерируются графические элементы, имитирующие поля, в количестве k.
4. При нажатии на какое-либо из полей появляется список характеристик выделенного поля.
5. В ячейки пользователем вносятся необходимые ему изменения. Для того, чтобы сохранить их, требуется нажать на кнопку сохранить.
6. Пользователь может продолжать выполнять действия 3-5 до тех пор, пока не убедится, что он заполнил все необходимые ему ячейки для каждого поля.
7. Для выхода из приложения необходимо нажать на кнопку «Выход».

## Графический интерфейс программы

При запуске программы пользователю представлено поле для ввода необходимого количества полей. После ввода ему нужно нажать на кнопку «Создать», чтобы сгенерировать поля.



Рисунок 1. Интерфейс программы при старте.

Примечание: если пользователь уже работал с приложением и сохранялся, то на экран сразу выведутся введенные им в процессе предыдущего сеанса работы поля.

После ввода количества полей (или при старте программы, в случае, если пользователь имеет сохранение) пользователю необходимо выбрать в списке то поле, которое он собирается редактировать.

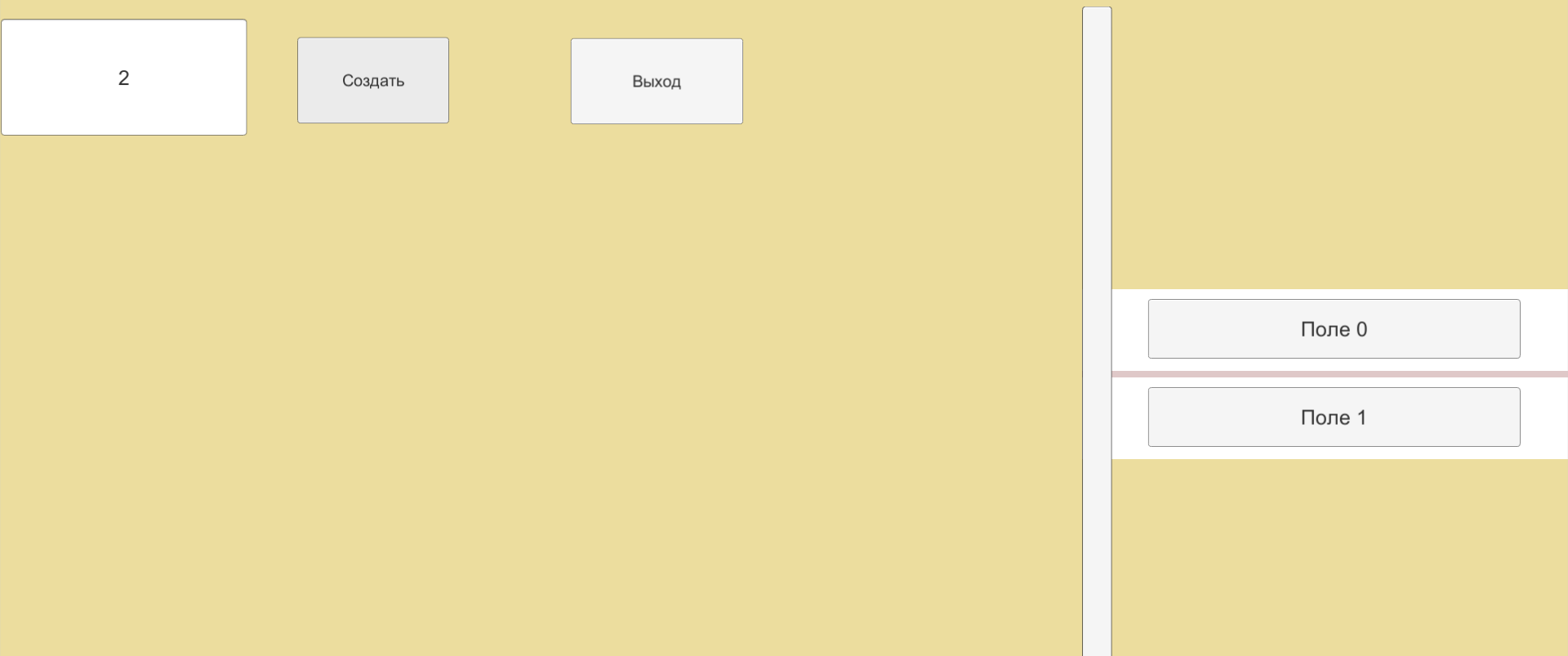


Рисунок 2. Иллюстрация работы программы после ввода количества полей

После нажатия на одно из полей пользователю предоставляется возможность изменить любое из возможных свойств, всего свойств пять: наименование растения, сорт, урожай, обработка от паразитов, подкармливание. При изменении последних трех свойств (при нажатии на кнопки с соответствующими названиями) появляются три дополнительные кнопки (высоко, средне, низко), взаимодействие с которыми также осуществляется посредством нажатия. Если была выбрана красная кнопка, то цвет кнопки-родителя (кнопки, на которую нажали перед тем, как появились дополнительные кнопки) поменяется на красный. Если была выбрана оранжевая, то на оранжевый и так далее. После нажатия на одну из дополнительных кнопок они все исчезают.

По завершению изменений пользователю необходимо нажать на кнопку с названием «Сохранить», чтобы сохранить внесенные им изменения для данного поля. Если не нажать кнопку «Сохранить», то все измененные пользователем поля заменяются на значения по умолчанию. Для завершения работы программы необходимо нажать на кнопку «Выход».

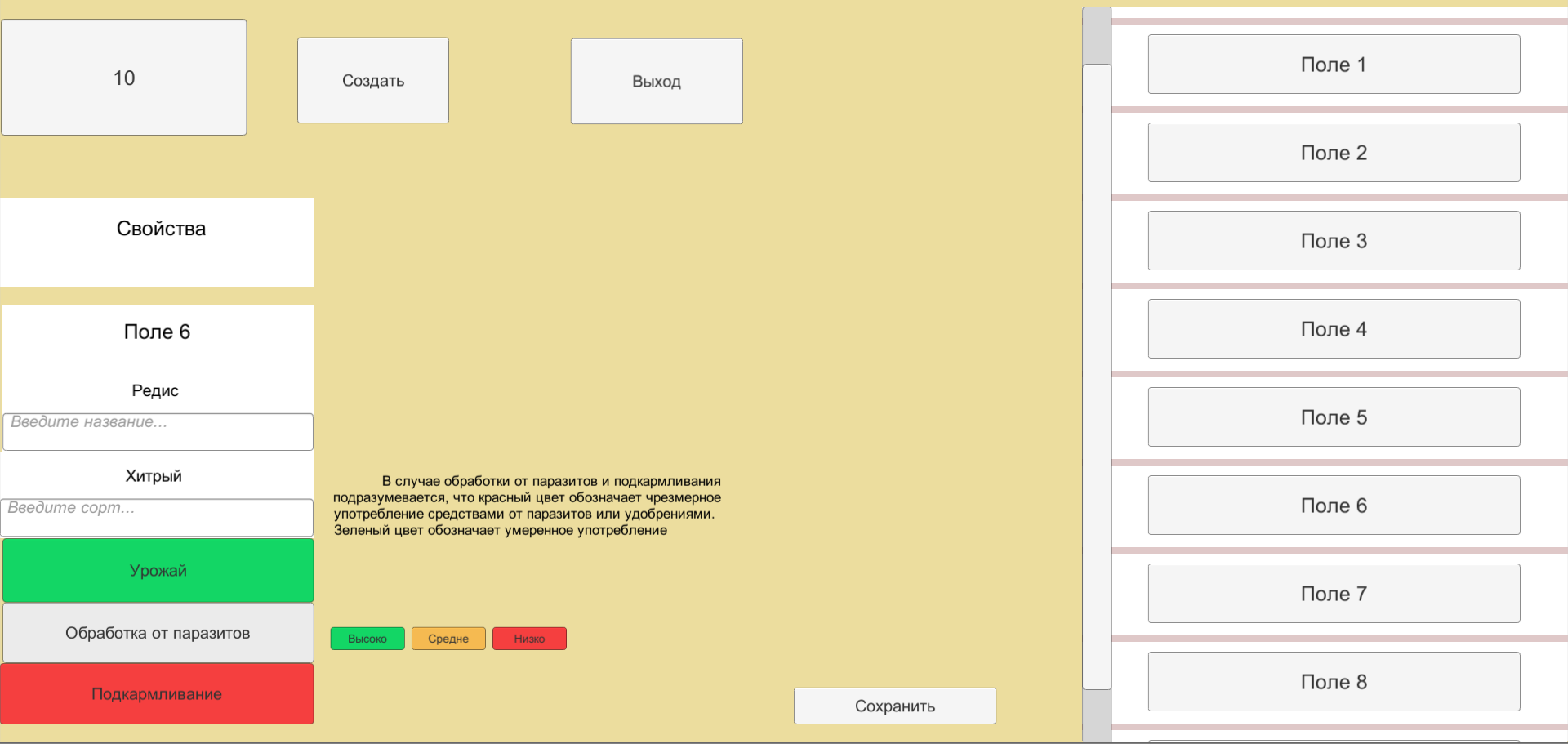


Рисунок 3. Иллюстрация работы программы после нажатия на одно из полей.

## Программный код

Данный код имеет три основных класса:

1. ScrollViewAdapter (рисунок 7) — это главный класс в программе, изначально предназначенный для графического отображения списка полей, который можно проматывать за ползунок. Графическое отображение происходит путем создания копий префаба поля (изображений, к которым прикреплена кнопка с названием поля) в том количестве, которое ввел пользователь. Класс содержит в себе:
   1. Поля, связывающие визуальные компоненты программы и программный код. Так как все изначально созданные в Unity классы по умолчанию наследуют свойства и методы класса MonoBehaviour, то поля класса ScrollViewAdapter могут использовать методы и свойства класса платформы Unity MonoBehaviour, который содержит в себе свойства и методы для работы с визуальными компонентами. Визуальные компоненты на сцене (рисунок 4) связываются с полями класса ScrollViewAdapter через инспектор в Unity (рисунок 5).

|  |  |
| --- | --- |
| C:\Users\Meijin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Иерархия.png | C:\Users\Meijin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Инспектор.png |

Рисунок 4.Иерархия объектов сцены Рисунок 5. Инспектор

* 1. Несколько подклассов:
* Save – это подкласс основного класса, реализующий в себе блок хранения данных о каждой из характеристик выбранного поле.
* TestBedView – это подкласс, связывающий в конструкторе визуальный компонент шаблона модели поля путем передачи ссылки на объект методом Find.

* 1. Методы класса:
  + Метод UpdateBeds (рисунок 8, 9), вызывающийся после нажатия на клавишу «Создать». При старте метода предварительно очищаются все предыдущие записи полей, затем создается экземпляр поля, который инициализируется либо данными, внесенными пользователем в результате предыдущего сеанса работы, либо значениями по умолчанию. Метод содержит в себе также множество различных методов, иерархично передающихся друг другу в качестве параметров. В совокупности данные методы создают список полей, выводят его на экран, а также инициализируют поля класса Save значениями.
  + Метод PathWay – инициализирует переменную path путем к директории с JSON файлом, в котором хранятся данные о состоянии программы на момент последнего нажатия на кнопку «Сохранить».
  + Метод Awake – автоматически и одноразово вызывается при старте программы. Основным назначением данного метода является проверка наличия сохранения. Если сохранение есть, то вызывается метод, который скрывает блок с полем для ввода количества полей и кнопкой, отвечающей за генерацию списка полей.

Также отдельно хотелось бы выделить класс JsonUtility [15], который можно использовать в совокупности с классом File, подключив библиотеку System.IO. Благодаря классу JsonUtility, в Unity становится возможным запоминание состояния информации о программе на момент последнего сохранения. Для того, чтобы осуществить сохранение информации, необходимо воспользоваться методом WriteAllText класса File, который создаст новый файл или переопределит его и указать в качестве параметров путь к расположению файла и некоторые данные, конвертированные в формат JSON с помощью метода ToJson.



Рисунок 6. Использование класса JsonUtility в программе.

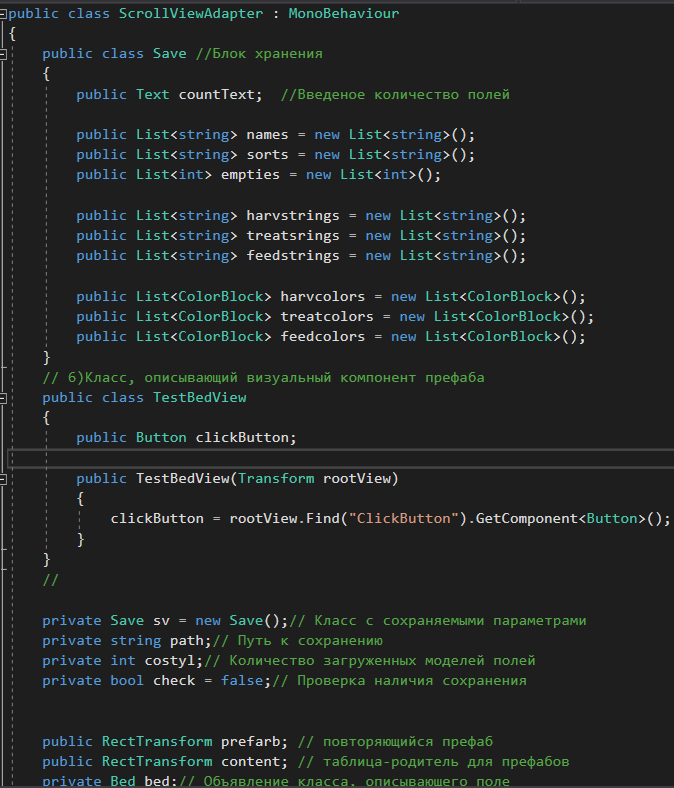


Рисунок 7. Класс ScrollViewAdapter.

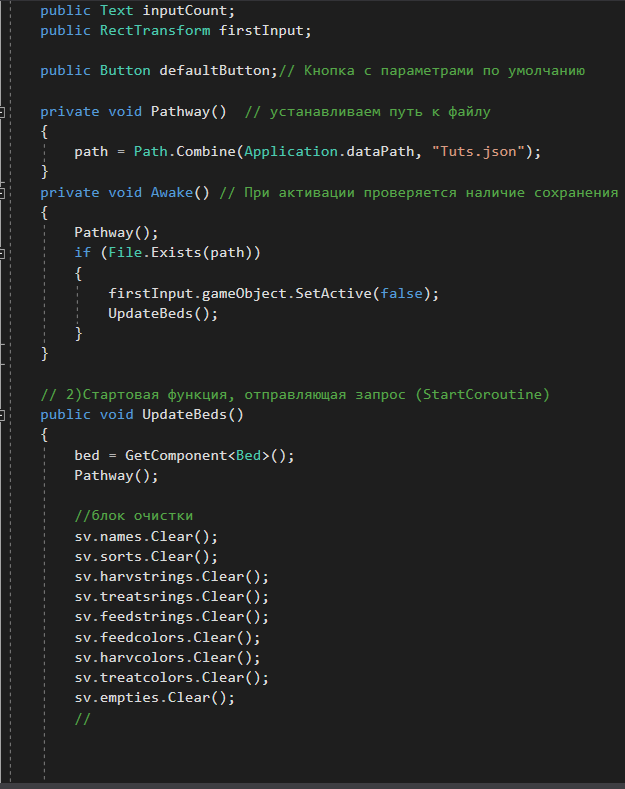


Рисунок 8. Начало описания функции UpdateBeds().

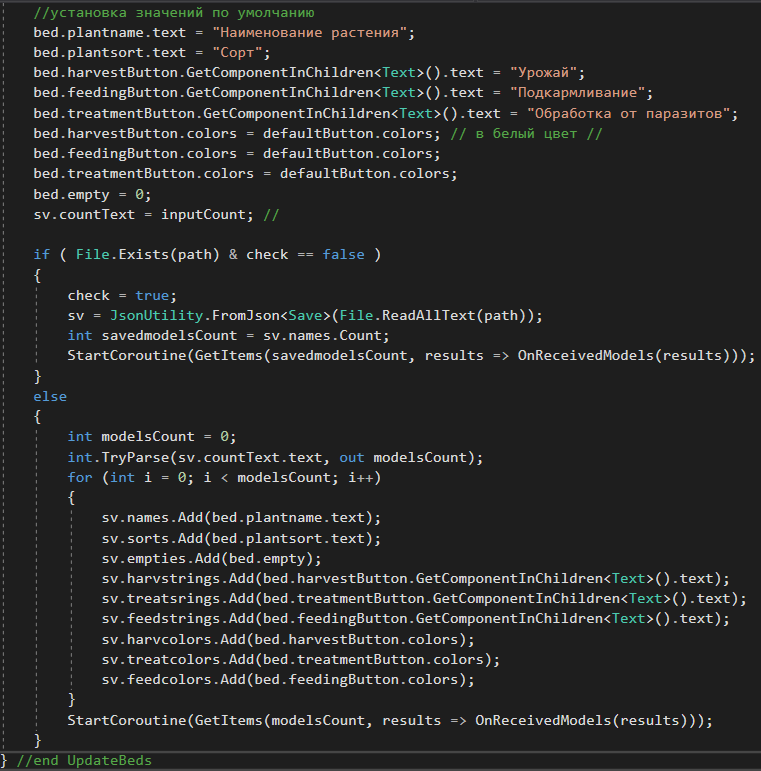


Рисунок 9. Завершение описания функции UpdateBeds().

1. Behaviour (рисунок 10) – это класс, описывающий поведение свойств полей. В нем содержатся методы, которые:
   1. Изменяют текст полей «Сорт» и «Наименование растения» на сцене при его вводе.
   2. Осуществляют выход из приложения.
   3. Осуществляют обзор нажатия на одну из кнопок «Урожайность», «Обработка» или «Удобрение», а затем и последующего взаимодействия с этими кнопками.

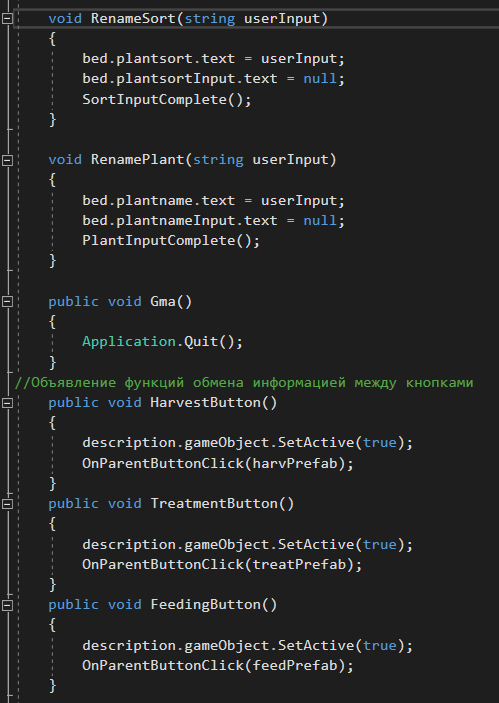


Рисунок 10. Основные методы класса Behaviour.

Также в данном классе хотелось бы выделить отдельно класс AllButtons (рисунок 11), в котором описаны поля, обозначающие «родительские» и «дочерние» кнопки на сцене. При этом в нем находится конструктор, назначением которого является инициализация полей данного класса по умолчанию при создании его экземпляра.

Инициализация «родительских кнопок» происходит путем присвоения компонента префаба-родителя типа RectTransform полю класса (префаб, в данном контексте – это контейнер, прикрепленный к «родительской» кнопке, который хранит в себе все «дочерние кнопки». Он должен передаться в качестве параметра при вызове конструктора). Инициализация «дочерних» кнопок происходит через метод Find, который проводит поиск дочерних объектов по имени. Основная функция данного класса заключается в том, что с помощью него ссылки на объекты сцены присваиваются непосредственно через код, а не через инспектор. Это необходимо делать для того, чтобы не перегружать инспектор излишней информацией.

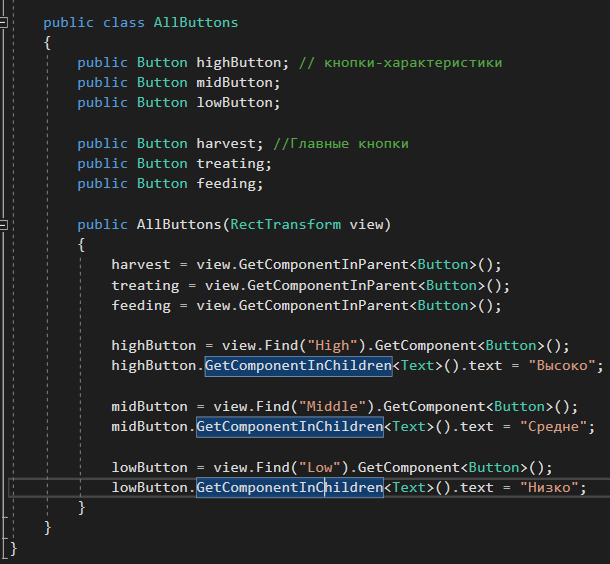


Рисунок 11. Класс AllButtons.

1. Bed (рисунок 12) – данный класс описывает шаблон поля земельного участка. Экземпляры данного класса присутствуют в обоих предыдущих классах, они необходимы для осуществления взаимодействия, так как в поля (поля класса) через инспектор присвоены ссылки на объекты на сцене, отвечающие за свойства поля (поля земельного участка).

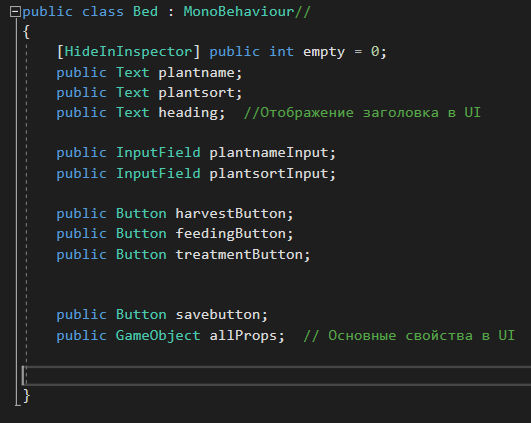


Рисунок 12. Поля класса Bed.

Дополнительно хотелось бы выделить метод SetActive. Данный метод применяется к объектам на сцене в том случае, когда необходимо их скрыть или вновь сделать видимыми. В данной работе, с помощью него, например, пользователь не может вводить данные о поле до тех пор, пока не введет количество полей. Таким образом, пользователю блокируется возможность получения преждевременного доступа к вводу данных о поле.

Также существует возможность скрывать / делать активными объекты на сцене непосредственно в самом редакторе Unity (рисунок 13, 14). Для этого в инспекторе необходимо снять / поставить отметку возле наименования объекта.

|  |  |
| --- | --- |
| C:\Users\Meijin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Set1.png |  |

Рисунок 13. Объект неактивен. Рисунок 14. Объект активен.

## Описание работы программы

В ходе разработки программы были произведены следующие улучшения:

1. Была произведена верстка визуальных объектов сцены, без которой объекты меняли свое расположение при изменении размера окна приложения.
2. Была исправлена ситуация, когда при генерации нового количества полей на сцене по-прежнему оставался блок, в котором изменялись свойства в поле, которое было нажато последним. Улучшение было произведено путем вызова метода SetActive (false) (рисунок 15) сразу после вызова метода UpdateBeds.

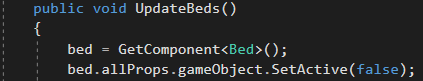


Рисунок 15. Вызов метода SetActive, скрытие блока со свойствами поля на сцене.

1. Была добавлена кнопка «Выход», отвечающая за выход из приложения.

Финальное тестирование: Смоделируем работу приложения. Сначала, на вход в большую ячейку с текстом «Введите количество полей» подается число 6, обозначающее количество полей земельного участка (рисунок 16). Затем, необходимо нажать на кнопку «Создать». В результате, в списке должно появиться 6 моделей поля (рисунок 17). После нажатия на одну из них должны появиться свойства выбранного поля. Здесь пользователь может ввести необходимые ему изменения для каждого свойства, после чего для того, чтобы программа запомнила внесенные данные, необходимо нажать на кнопку «Сохранить» (рисунок 18). После того, как пользователь произведет предыдущую операцию для каждого из полей, он может выйти из приложения путем нажатия на кнопку «Выход». При повторном запуске приложение должно воспроизвести состояние программы на момент последнего нажатия на кнопку «Сохранить» (рисунок 19).

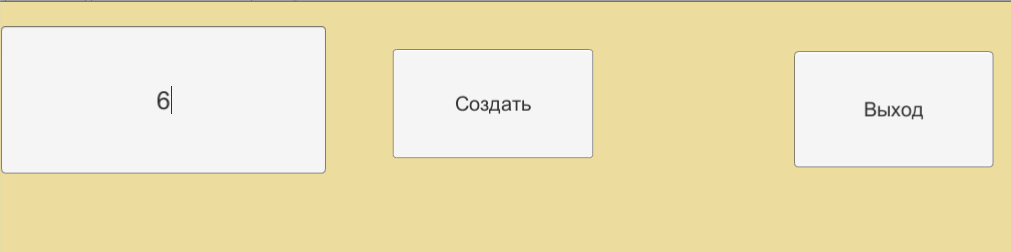


Рисунок 16. На вход подается число 6.

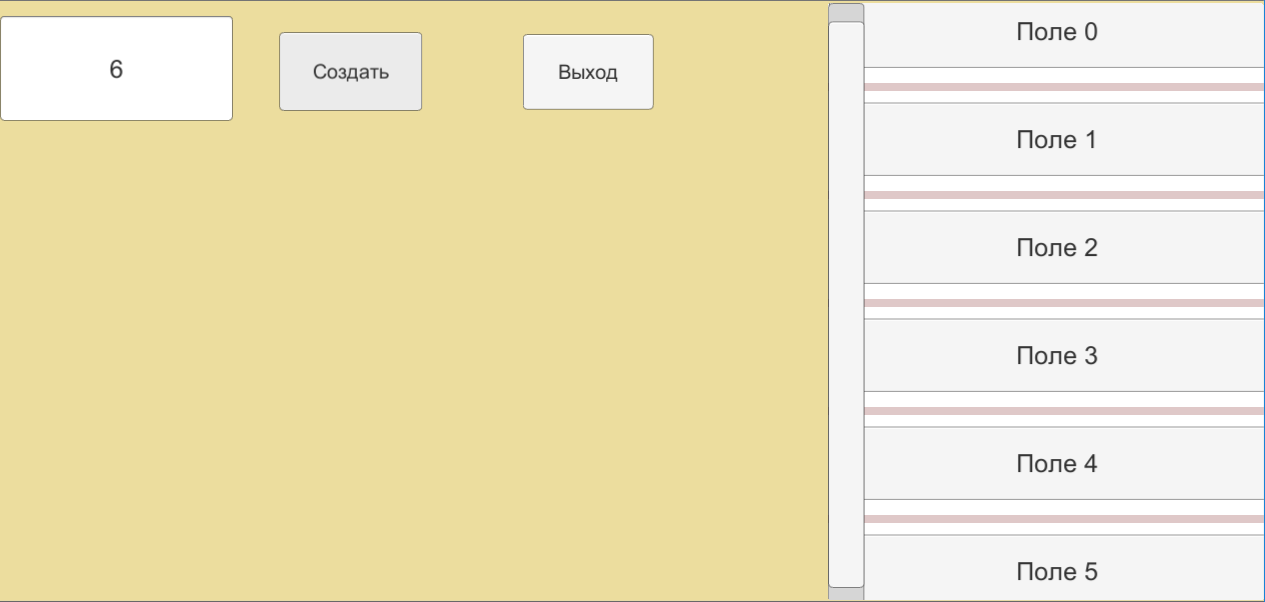


Рисунок 17. После нажатия на кнопку «Создать» появляется список с 6 полями.

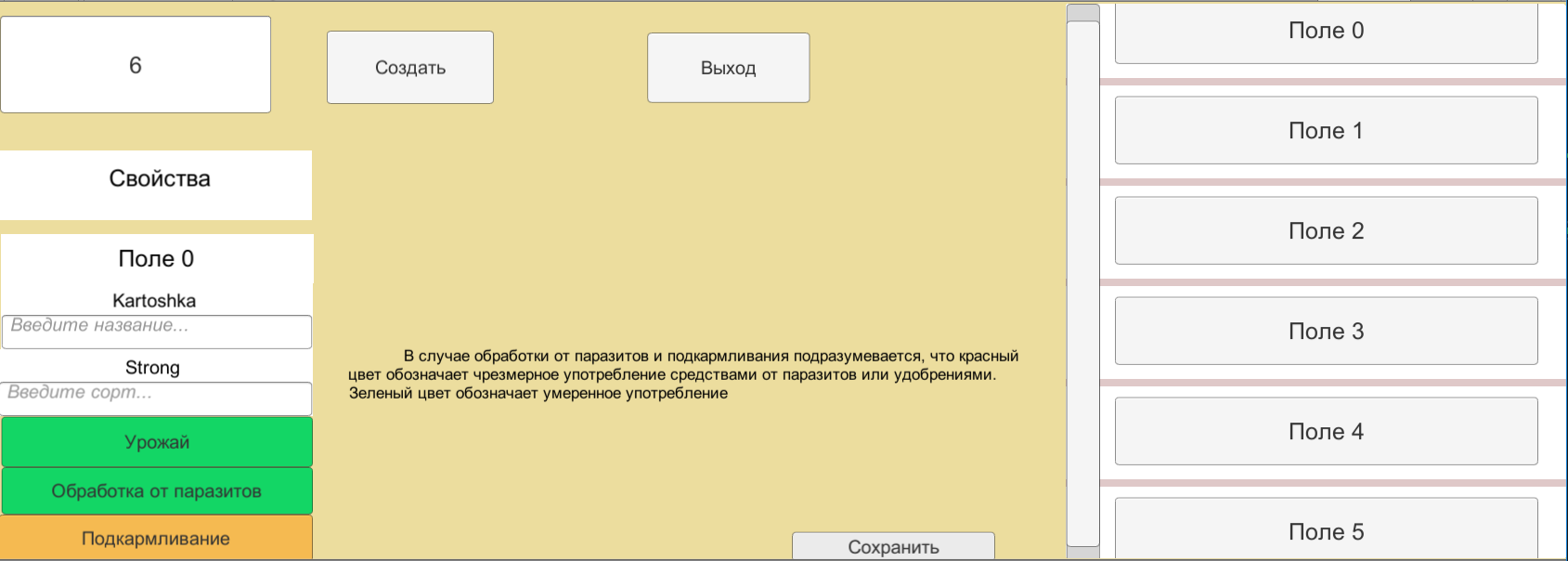


Рисунок 18. Изменение свойств, их сохранение путем нажатия на кнопку «Сохранить».

Рисунок 19. Успешное сохранение информации.



В ходе тестирования были проверены следующие элементы, отвечающие за ввод:

1. Ячейка, в которую необходимо внести количество грядок (полей). При внесении цифры меньше нуля или символьного значения программа работает исправно, пользователю необходимо ввести значение заново.
2. Ячейки, в которые вводятся сорт и наименование растения. В них возможно внести любые данные, значение ячеек определяется самим пользователем.
3. Ячейки, в которые вводятся оценочные характеристики о свойствах «Урожай», «Обработка от паразитов», «Подкармливание» путем изменения цвета данных ячеек (рисунок 20). Ячейки принимают только значения цвета кнопок, появляющихся при нажатии на саму ячейку.



Рисунок 20. Изменение цвета ячейки.

В ходе разработки программы возникло три основных проблемы:

1. Проблема разбиения программного кода на отдельные скрипты [19], которые должны отвечать за отдельные функции кода, взаимодействие между этими скриптами. Так как в Unity нельзя создавать экземпляры класса, наследующие класс MonoBehaviour, который является необходимым для работы с визуальными объектами, то нельзя воспользоваться словом new для создания нового экземпляра класса. Проблема была решена вызовом метода GetComponent [18] (Рисунок 9.1), который возвращает необходимый тип, таким образом, заменяя привычное создание объекта класса с помощью конструкции «Тип Наименование\_экземпляра = new Тип()».
2. Ошибка NullReferenceException, которая, как оказалось, была связана с тем, что метод GetComponent возвращает Null в случае, если скрипт, в котором описан класс, тип которого необходимо получить с помощью GetComponent, и скрипт, в котором вызывается метод GetComponent не привязаны к одному и тому же объекту. Проблема была решена путем связывания скриптов к одному объекту на сцене.
3. Проблема сохранения результатов работы. Это произошло из-за того, что программа должна сохранять относительно большие массивы данных, а не несколько значений. Проблема была решена путем использования класса JsonUtility, а также путем создания отдельного класса Save, назначением которого является хранение данных о свойствах всех полей в динамических массивах. Таким образом, при повторном запуске программа возвращает поля во введенном в прошлом сеансе работы количестве, а также она сохраняет введенные до этого свойства, следовательно, программа успешно сохраняет необходимую для пользователя информацию.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнении курсовой работы при изучении теории были рассмотрены общие принципы разработки приложений на платформе Unity, а также принципы программирования.

При выполнении практической части курсовой работы было освоено множество различных технологий программирования Unity на языке C#, в частности, технология построения связи между визуальным объектом на сцене и программным кодом.

Было разработано Windows - приложение Гор\_город, позволяющее пользователю запоминать месторасположение посаженных им культур на полях, а также, запоминающие свойства данных полей. Следственно, приложение реализует все поставленные на этапе разработки задачи. Также был проведен анализ аналогичных приложений, который показал, что разрабатываемое приложение не имеет абсолютных аналогов, а также то, что данное ПО действительно может быть полезно для людей, связанных с сельскохозяйственной отраслью.

Приложение было протестировано, и, на основе полученных результатов была проведена отладка, которая показала, что приложение работает корректно.

Таким образом, все задачи, поставленные в курсовой работе, были выполнены, а цель работы – разработка приложения «Гор\_город», создающего визуальную модель земельного участка – достигнута.

# **БИБЛИОГРАФИЯ**

1. Разработка программного обеспечения: этапы и принципы https://habr.com/ru/ company/edison/blog/267671/ (дата обращения: 10.12.19).
2. Выбор технологии для большого и не очень проекта https://habr.com/ ru/company/SECL\_GROUP/blog/315734/ (дата обращения: 10.12.19).
3. База данных (БД) https://hostiq.ua/wiki/database/ (дата обращения: 10.12.19).
4. ФЗ «Об информации, информационных технологиях и о защите информации» http://www.consultant.ru/document/cons\_doc\_LAW\_61798/ (дата обращения: 10.12.19).
5. Земельный кодекс Российской Федерации http://www.consultant.ru/document/ cons\_doc\_LAW\_33773/ (дата обращения: 10.12.19).
6. Unity – Manual: Unity User Manual (2019.2) https://docs.unity3d.com /Manual/index.html/
7. Руководство по программированию на C# | Microsoft Docs <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/> (дата обращения: 10.12.19).
8. Общие сведения о платформе .NET Framework // Microsoft URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview/ (дата обращения: 10.12.19).
9. Что такое объектно-ориентированное программирование https://younglinux. info/oopython/oop.php/ (дата обращения: 10.12.19).
10. Пользовательский интерфейс | Новичку https://www.internet-technologies.ru /articles/polzovatelskiy-interfeys.html (дата обращения: 10.12.19).
11. ГОСТ 34.601-90 Информационная технология. http://docs.cntd.ru/document/gost-34-601-90 (дата обращения: 10.12.19).
12. GUI (Graphical User Interface). https://ru.bmstu.wiki/GUI\_(Graphical\_User\_ Interface) (дата обращения: 10.12.19).
13. Среды разрабоки программного обеспечения http://nit.miem.edu.ru/sbornik/2009/ plen/008.html/ (дата обращения: 10.12.19).
14. Отладка программы: 3 типа ошибок https://proglib.io/p/debugging (дата обращения: 10.12.19).
15. JSON Serialization - Unity – Manual https://docs.unity3d.com/Manual/JSON Serialization.html (дата обращения: 10.12.19).
16. Я не знаю ООП / Хабр – Habr https://habr.com/ru/post/147927/ (дата обращения: 10.12.19).
17. Основные принципы ООП: https://gos-it.fandom.com/wiki/ (дата обращения: 10.12.19).
18. API скриптов: GameObject.GetComponent – Unity https://docs.unity3d.com/ru /current/ScriptReference/GameObject.GetComponent.html (дата обращения: 10.12.19).
19. Руководство: Создание и Использование Скриптов – Unity https:// docs.unity3d.com/ru/current/Manual/CreatingAndUsingScripts.html (дата обращения: 10.12.19).
20. Герберт Шилдт - C# 4.0: полное руководство https://www.libfox.ru/599452-gerbert-shildt-c-4-0-polnoe-rukovodstvo.html (дата обращения: 10.12.19).