**Частное учреждение образования «Минский колледж предпринимательства»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | УТВЕРЖДАЮ |
|  |  |  |  | Заведующий отделением ПОИТ |
|  |  |  |  |  |
|  |  |  |  | М.В. Меньшикова |
|  |  |  |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г. |

**Игра «Shop Town»**

**Пояснительная записка**

**МКП.3170814-01 81 15**

*technicaldocs.ru*

*Подпись и дата*

*Взам. инв. №*

*Инв. № дубл.*

*Инв. № подл.*

*Подпись и дата*

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | Учащийся |
|  |  |  |
|  |  | Л.Л. Татур |
|  |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г. |
|  |  |  |
|  |  | Руководитель |
|  |  |  |
|  |  | С.Н. Науменко |
|  |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

2021

**АННОТАЦИЯ**

Курсовой проект: 56с., 28 рис., 7 источников.

JSON, SHOP, ПРИЛОЖЕНИЕ, C#, ПРОГРАММИРОВАНИЕ, UNITY, SCRIPTS

Объектом и предметом исследования является среда разработки Unity и её безграничные возможности.

Цель проекта: создать игру для веселого времяпровождения и получить опыт разработки в Unity.

В процессе работы выполнены следующие исследования: построена структура для сериализации данных, разработано приложение под ОС Windows, улучшен уровень объема профессионально значимых знаний, умений и навыков.

Автор подтверждает, что приведенный в работе расчетно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические, методологические и методические положения и концепции сопровождаются ссылками на их авторов.**ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ**

API – Application Programming Interface.

ПП – программный продукт.

ЛКМ – левая кнопка мыши.

ПКМ – правая кнопка мыши.

JSON – JavaScript Object Notation.

**МИНСКИЙ КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА**

(название учреждения образования)

УТВЕРЖДАЮ

Председатель цикловой (предметной)   
комиссии «Программное обеспечение информационных технологий»

/ Ю.А. Доманова /

« » 2021 г.

**ЗАДАНИЕ**

**на курсовой проект**

Учащейся Татур Любови Леонидовне

(фамилия, имя, отчество)

курса 3 группы П-1807

по учебной дисциплине «Конструирование программ и языки программирования»

Тема курсового проекта (работы) «Игра «Town Shop»

Исходные данные: данные, получаемые от пользователя; данные, хранящиеся в сериализованных файлах.

**Состав проекта (работы)**

|  |  |
| --- | --- |
| **Пояснительная записка** | |
| Содержание разделов | Срок выполнения |
| Введение. Назначение и область применения | 01.02.21 – 14.02.21 |
| Постановка задачи. Описание организации данных | 15.02.21 – 28.02.21 |
| Описание программных средств. Проектирование интерфейса | 01.03.21 – 14.03.21 |
| Кодирование. Описание логической структуры программы | 01.03.21 – 04.04.21 |
| Тестирование и анализ полученных результатов | 05.04.21 – 11.04.21 |
| Руководство по использованию программного средства | 12.04.21 – 18.04.21 |
| Заключение. Источники, использованные при разработке | 19.04.21 – 25.04.21 |
| Приложения |  |

**Графическая часть проекта**

Лист – 1\_\_\_ER-диаграмма

Лист – 2 Диаграмма вариантов использования

Лист – 3\_\_\_Диаграмма классов

Дата выдачи «01» февраля 2021 г.

Срок сдачи «26» апреля 2021 г.

**Преподаватель-руководитель** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**курсового проекта**

**Подпись учащегося**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc70631008)

[1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 8](#_Toc70631009)

[1.1 Обоснование необходимости автоматизации 8](#_Toc70631010)

[1.2 Определение данных и их представление 9](#_Toc70631011)

[2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 11](#_Toc70631012)

[2.1 Постановка задачи 11](#_Toc70631013)

[2.2 Описание организации данных 11](#_Toc70631014)

[2.3 Описание программных средств 11](#_Toc70631015)

[2.4 Проектирование интерфейса 14](#_Toc70631016)

[2.5 Описание логической структуры программы 20](#_Toc70631017)

[3 ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫЗ РЕЗУЛЬТАТОВ 23](#_Toc70631018)

[4 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА 33](#_Toc70631020)

[ЗАКЛЮЧЕНИЕ 37](#_Toc70631021)

[ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЯ: 38](#_Toc70631022)

[ПРИЛОЖЕНИЯ 39](#_Toc70631023)

# ВВЕДЕНИЕ

Видеоигра — игра с использованием изображений, сгенерированных электронной аппаратурой. Другими словами, видеоигра является электронной игрой, которая базируется на взаимодействии человека и устройства посредством визуального интерфейса, например, телевизора, монитора компьютера или телефона.

Сейчас известно 7 типов компьютерных игр:

1) Квесты — осуществление одного или нескольких персонажей к конечной цели, осуществляя преодоление препятствий;

2) Экшн — это игры от первого лица. Говоря простым языком: "бродишь и стреляешь";

3) Ролевые игры;

4) Стратегии — в игре такого жанра отсутствует отдельный персонаж. Там человек управляет определенными внутриигровыми процессами. Сначала используя логику, игрок выполняет свою логическую цепочку решения, а после наблюдает последствия своих действий;

5) Симуляторы — достаточно популярный жанр. Суть его в том, что он имитирует, например, управление на автомобиле, самолете, корабле, космическом аппарате. В таких играх нет сюжета, они не развивают какого-то привыкания;

6) Логические игры;

7) Азартные игры — идет направленность на мозг человека. Такой вид игры развивает мыслительную деятельность, подводит человека к собственному логическому размышлению, заставляет креативно мыслить, и строить логические цепочки.

Основной целью выполнения курсового проекта является разработка программного средства для занимательного времяпровождения и отдыха, а также расширение, углубление знаний по дисциплине «конструирование программ и языки программирования». Получение навыков проектирования и создания программного обеспечения.

Задачи выполнения курсового проекта:

- систематизация полученных знаний;

- углубление уровня и расширение объема профессионально значимых знаний, умений и навыков;

- формирование умений и навыков самостоятельной работы;

- практическое использование сериализации для хранения информации;

- разработка интуитивно простого интерфейса для пользователя.

# 1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

## 1.1 Обоснование необходимости автоматизации

Microsoft Visual Studio – это интегрированная среда разработки от Microsoft. Он используется для разработки компьютерных программ для Microsoft Windows, а также веб-сайтов, веб-приложений и веб-сервисов.

Visual Studio использует платформы разработки программного обеспечения Microsoft:

– Windows API;

– Windows Forms;

– Windows Presentation Foundation;

– Windows Store;

– Microsoft Silverlight.

Visual Studio включает в себя редактор кода, поддерживающий IntelliSense, а также рефакторинг кода. Интегрированный отладчик работает как отладчик уровня источника и отладчик уровня машины.

Другие встроенные инструменты включают конструктор форм для создания приложений с графическим интерфейсом, веб-дизайнер, дизайнер классов и конструктор схем базы данных. Он принимает плагины, которые расширяют функциональные возможности практически на каждом уровне, включая добавление поддержки для систем контроля версий (таких как Subversion) и добавление новых наборов инструментов, таких как редакторы и визуальные дизайнеры для языков, специфичных для предметной области, или наборов инструментов для других аспектов жизненного цикла разработки программного обеспечения.

Одним из аналогов Visual Studio является Eclipse. Eclipse, отстаёт от Visual Studio. Eclipse имеет ряд минусов:

* при недостаточном выделении памяти для приложения, он будет работать медленно.
* при запуске Eclipse в проекте, хранящемся в сетевой файловой системе, он будет работать медленнее.
* при запуске Eclipse из сетевой установки, она будет работать медленнее.

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

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями.

## 1.2 Определение данных и их представление

Входными данными для разработки ПП являются:

* данные, получаемые от пользователя;
* данные, хранящиеся в сериализованных файлах;

Выходными данными разрабатываемого ПП являются:

* информация, сохраняемая в сериализованных файлах.

Разрабатываемый ПП должен создавать и модифицировать следующие данные в сериализованных файлах (рис 1.1):

* количество игровой валюты;
* содержание инвентаря пользователя;
* положение расставленных в игровой комнате предметов.

****

Рис. 1.1. Информация, хранящаяся в сериализованных файлах

# 2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

## 2.1 Постановка задачи

Разработать проект, который будет интересен пользователям, в целях занимательного времяпровождения и получения положительных эмоций.

Приложение «Shop Town» предлагает пользователю получить игровую валюту, играя в интерактивную мини-игру, и тратить её на интересующие товары, которые пользователь сможет разместить в своем игровом доме.

На данный момент времени существуют подобные приложения, однако игра «Shop Town» имеет возможность дальнейшего совершенствования, таких, как добавления новых различных игр, предметов, иных функций.

Разработанное приложение содержит следующие функции: интерактивная мини-игра для получения игровой валюты, покупка предметов в магазинах, расставление предметов из инвентаря в игровую комнату, возврат предмета из комнаты в инвентарь, продажа предметов из инвентаря.

## 2.2 Описание организации данных

Данные, хранящиеся в сериализованном файле, диаграмма сущность-связь которой представлена в приложении 1:

1)Количество игровой валюты;

2) Информация о предметах, хранящихся в инвентаре пользователя;

3) Информация о предметах, расставленных в игровом доме.

## 2.3 Описание программных средств

Проект был разработан в программе Unity, которая имеет свои достоинства и недостатки.

Как правило, игровой движок предоставляет множество функциональных возможностей, позволяющих их задействовать в различных играх, в которые входят моделирование физических сред, карты нормалей, динамические тени и многое другое. В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие визуальной среды разработки и межплатформенная поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и интегрированную среду, цепочку сборки, что направлено на повышение производительности разработчиков, в частности, этапов создания прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места развертывания (установка на персональном компьютере, на мобильном устройстве, консоли и т. д.), но и наличие инструментария разработки (интегрированная среда может использоваться под Windows и Mac OS).

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов наследования, объекты в Unity создаются посредством объединения функциональных блоков, а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр.

В качестве недостатков приводятся ограничение визуального редактора при работе с многокомпонентными схемами, когда в сложных сценах визуальная работа затрудняется. Вторым недостатком называется отсутствие поддержки Unity ссылок на внешние библиотеки, работу с которыми программистам приходится настраивать самостоятельно, и это также затрудняет командную работу. Ещё один недостаток связан с использованием шаблонов экземпляров (англ. prefabs). С одной стороны, эта концепция Unity предлагает гибкий подход визуального редактирования объектов, но с другой стороны, редактирование таких шаблонов является сложным. Также, WebGL-версия движка, в силу специфики своей архитектуры (трансляция кода из C# в С++ и далее в JavaScript), имеет ряд нерешённых проблем с производительностью, потреблением памяти и работоспособностью на мобильных устройствах.

Проект был разработан с помощью Visual Studio на языке С#, который имеет свои достоинства и недостатки.

Плюсы Visual Studio:

* бесплатно распространяется Visual Studio Community, с достаточным набором возможностей.
* платные версии могут предоставляться учебным заведениям и студентам бесплатно.
* удобная система умного автодополнения.
* большое количество настроек среды разработки под «себя», благодаря встроенным механизмам и доступным дополнениям.

Минусы Visual Studio:

* версии Visual Studio Professional и Visual Studio Enterprise являются платными.
* тяжеловесная IDE, достаточно требовательная к железу.
* ряд разработчиков считают ее избыточной для создания мелких проектов.
* достаточно маленькая скорость запуска проектов и приложений.
* не работает на Linux.
* платные версии с расширенным функционалом распространяются по подписке: Professional – от 45$ в месяц, Enterprise – от 250$ в месяц.

## 2.4 Проектирование интерфейса

Курсовой проект состоит из 6 сцен:

Сцена 1 – комната пользователя (рис. 2.1);

Сцена 2 – улица с магазинами (рис. 2.2);

Сцена 3 – магазин мебели (рис. 2.3);

Сцена 4 – магазин ванных принадлежностей (рис. 2.4);

Сцена 5 – магазин стикеров (рис. 2.5);

Сцена 6 – центр игровых автоматов (рис. 2.6).

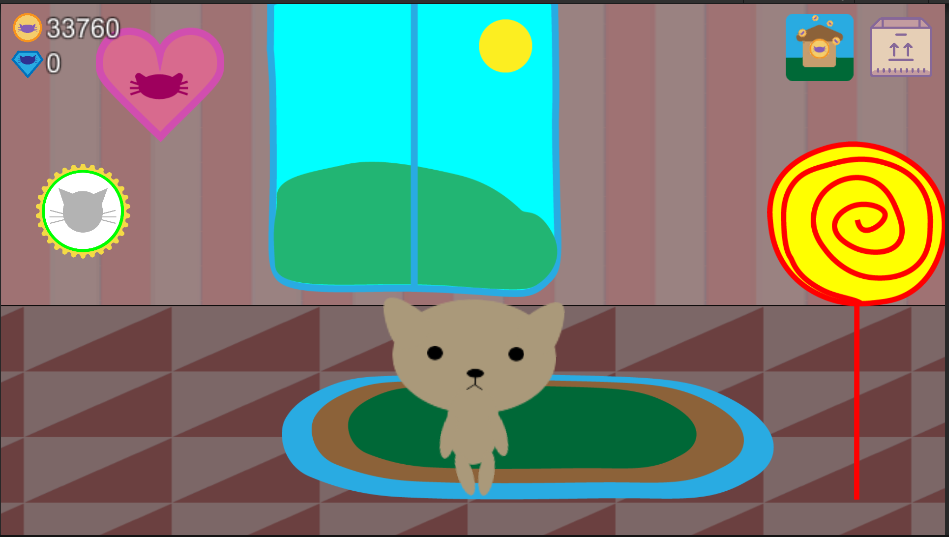


Рис. 2.1. Комната пользователя

Таблица 2.1 – Компоненты сцены «Комната пользователя»:

|  |  |
| --- | --- |
| Название компонента | Назначение компонента |
| cat | Изображение персонажа |
| bg | Задний фон |
| Main Camera | Главная камера |
| coin | Отображение количества монет |
| Button1, Button2 | Кнопки, осуществляющие переключения между сценами |
| Inventory\_tab | Отображение инвентаря |
| sellingPanel | Панель для продажи предметов |

** Рис. 2.2. Улица с магазинами

Таблица 2.2 – Компоненты сцены «Улица с магазинами»:

|  |  |
| --- | --- |
| Название компонента | Назначение компонента |
| cat | Изображение персонажа |

*Продолжение таблицы 2.2*

|  |  |
| --- | --- |
| bg | Задний фон |
| Main Camera | Главная камера |
| coin | Отображение количества монет |
| Button1 | Кнопкa, осуществляющая переключение между сценами |
| Shop1,Shop2 … Shop5 | Кнопкa, осуществляющая переход в сцену определенного магазина |

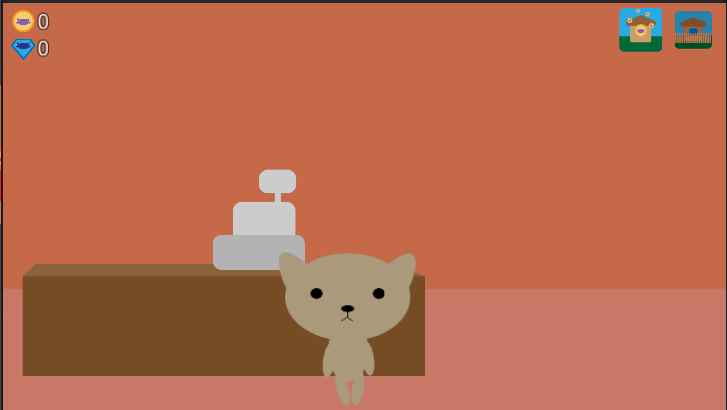
**

Рис. 2.3. Магазин мебели

Таблица 2.3 – Компоненты сцены «Магазин мебели»:

|  |  |
| --- | --- |
| Название компонента | Назначение компонента |
| cat | Изображение персонажа |

*Продолжение таблицы 2.3*

|  |  |
| --- | --- |
| bg | Задний фон |
| Main Camera | Главная камера |
| coin | Отображение количества монет |
| Button1, Button2 | Кнопки, осуществляющие переключения между сценами |
| items | Компонент, имеющий в себе определенное количество компонентов товаров |
| buyingPanel | Панель для покупки предметов |

**

Рис. 2.4. Магазин ванных принадлежностей

Таблица 2.4 – Компоненты сцены «Магазин ванных принадлежностей»:

|  |  |
| --- | --- |
| Название компонента | Назначение компонента |
| cat | Изображение персонажа |
| bg | Задний фон |

*Продолжение таблицы 2.4*

|  |  |
| --- | --- |
| Main Camera | Главная камера |
| coin | Отображение количества монет |
| Button1, Button2 | Кнопки, осуществляющие переключения между сценами |
| items | Компонент, имеющий в себе определенное количество компонентов товаров |
| byuingPanel | Панель для покупки предметов |

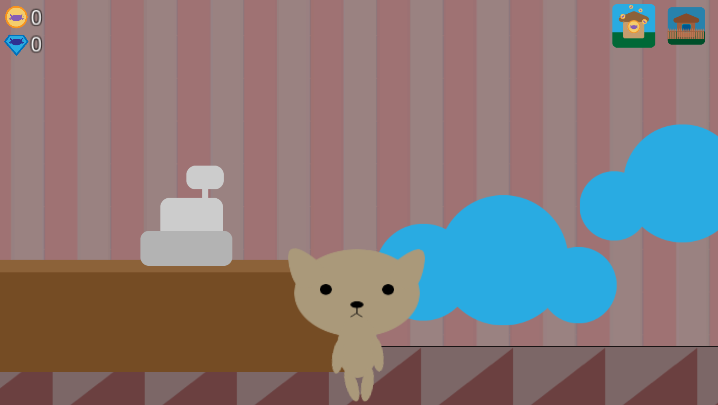
**

Рис. 2.5. Магазин стикеров

Таблица 2.5 – Компоненты сцены «Магазин стикеров»:

|  |  |
| --- | --- |
| Название компонента | Назначение компонента |
| cat | Изображение персонажа |
| bg | Задний фон |
| Main Camera | Главная камера |

*Продолжение таблицы 2.5*

|  |  |
| --- | --- |
| coin | Отображение количества монет |
| Button1, Button2 | Кнопки, осуществляющие переключения между сценами |
| items | Компонент, имеющий в себе определенное количество компонентов товаров |
| byuingPanel | Панель для покупки предметов |

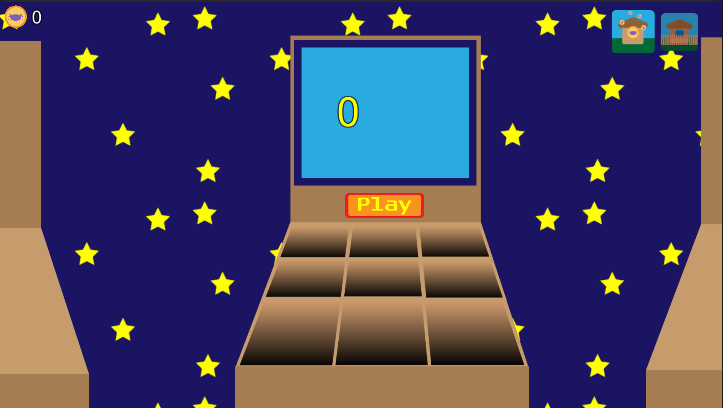
**

Рис. 2.6. Центр игровых автоматов

Таблица 2.6 – Компоненты сцены «Центр игровых автоматов»:

|  |  |
| --- | --- |
| Название компонента | Назначение компонента |
| bg | Задний фон |
| Main Camera | Главная камера |
| coin | Отображение количества монет |
| mainPlayPlace | Основная игровая панель |

## 2.5 Описание логической структуры программы

Приложение состоит из 17 классов. Диаграмма классов представлена в приложении 3.

Методы класса skriptForDeleteObj и их назначения указаны в таблице 2.7.

Таблица 2.7 – Методы класса skriptForDeleteObj

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| SavingFixing | Сохраняет позицию размещенного предмета в комнате в сериализованный файл. |

Методы класса MoveIntoShop и их назначения указаны в таблице 2.8.

Таблица 2.8 – Методы класса MoveIntoShop

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| OnMouseDown | Перемещает пользователя на сцену магазина. |

Методы класса GameScript и их назначения указаны в таблице 2.9.

Таблица 2.9 – Методы класса GameScript

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| GameOver | Обработка конца игры. |
| GameStep | Описание одного шага игры. |
| StartGame | Описание логики начала игры. |
| WasBonked | Обработка нажатия по котику. |

Методы класса ForCatsCollisions и их назначения указаны в таблице 2.10.

Таблица 2.10 – Методы класса ForCatsCollisions

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| OnMouseDown | Обработка нажатия по котику. |

Методы класса homeButton и их назначения указаны в таблице 2.11.

Таблица 2.11 – Методы класса homeButton

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| OnMouseDown | Перемещение пользователя домой. |

Методы класса SavingValues и их назначения указаны в таблице 2.12. Класс используется для сериализации.

Таблица 2.12 – Методы класса SavingValues

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| SavingValues | Конструктор |

Методы класса streetButton и их назначения указаны в таблице 2.13.

Таблица 2.13 – Методы класса streetButton

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| OnMouseDown | Перемещение пользователя на улицу. |

Методы класса SkriptPlayer и их назначения указаны в таблице 2.14.

Таблица 2.14 – Методы класса SkriptPlayer

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| Update | Перемещение игрока. |

Методы класса ItemSkripts и их назначения указаны в таблице 2.15.

Таблица 2.15 – Методы класса ItemSkripts

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| MoveObject | Перемещение предмета по экрану. |
| SelectObject | Выбрать предмет для перемещения. |

Методы класса inventory и их назначения указаны в таблице 2.16.

Таблица 2.16 – Методы класса inventory

|  |  |
| --- | --- |
| Наименование метода | Назначение метода |
| AddItem | Добавление предмета в инвентарь. |
| CansellSell | Обработка нажатия кнопки «Отмена» в панели продажи предмета. |
| ChangeAmount | Изменение количество предмета в инвентаре. |
| Save | Сохранение данных в сериализованный файл. |
| Load | Загрузка данных из сериализованного файла. |

# 3 ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫЗ РЕЗУЛЬТАТОВ

Метод «черного ящика». Основан на принципе «вход-выход». Программе подаются некоторые данные на вход и проверяются результаты, в надежде найти несоответствия. При этом как именно работает программа, считается несущественным.

Метод «белого ящика». В этом методе тестовые данные получают путем анализа логики программы.

Этапы процесса тестирования:

1. Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования системы.
2. Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, граничные объемы данных.
3. Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области. Программа должна отвергать любые данные, которые она не в состоянии обработать правильно. Наихудшая ситуация возникает тогда, когда программа принимает данные, выходящие за допустимые пределы и выдает неверный результат.

В проекте использовался метод тестирования «черного ящика». Программа была протестирована на следующих тестах:

1) Тест №1. Попытка разместить предмет в комнате.

2) Тест №2. Попытка вернуть предмет в инвентарь.

3) Тест №3. Попытка продать предмет.

4) Тест №4. Попытка попасть на улицу.

5) Тест №5. Попытка войти в магазин.

6) Тест №6. Попытка купить предмет.

7) Тест №7. Попытка купить предметов на сумму больше, чем имеется.

8) Тест №8. Попытка начать мини-игру.

9) Тест №9. Попытка попасть по котику во время мини-игры.

10) Тест №10. Попытка проиграть в мини-игре.

Тест №1. При открытом инвентаре нажимаем по предмету ЛКМ (в данном случае «Стикер суша», рис. 3.1). Тащим предмет на нужное место и нажимаем на ПКМ (рис. 3.2).

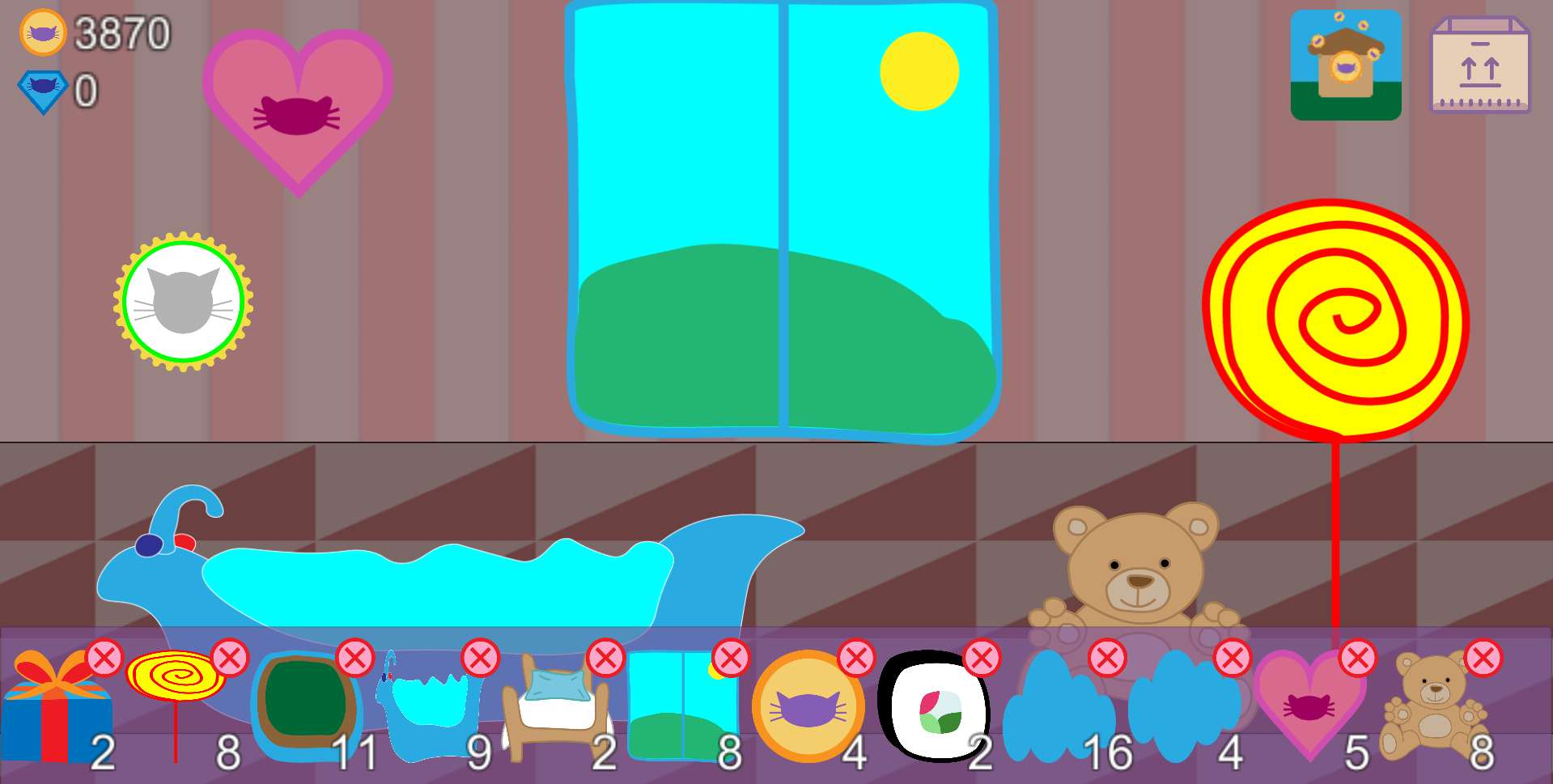


Рис. 3.1. Нажатие ЛКМ по предмету

# 

Рис. 3.2. Размещение предмета в комнате и нажатие ПКМ

Тест №2. При открытом инвентаре нажимаем по предмету в комнате ПКМ (в данном случае «Дневное окно», рис. 3.3). Предмет возвращается в инвентарь (рис. 3.4).



Рис. 3.3. Нажатие по предмету в комнате

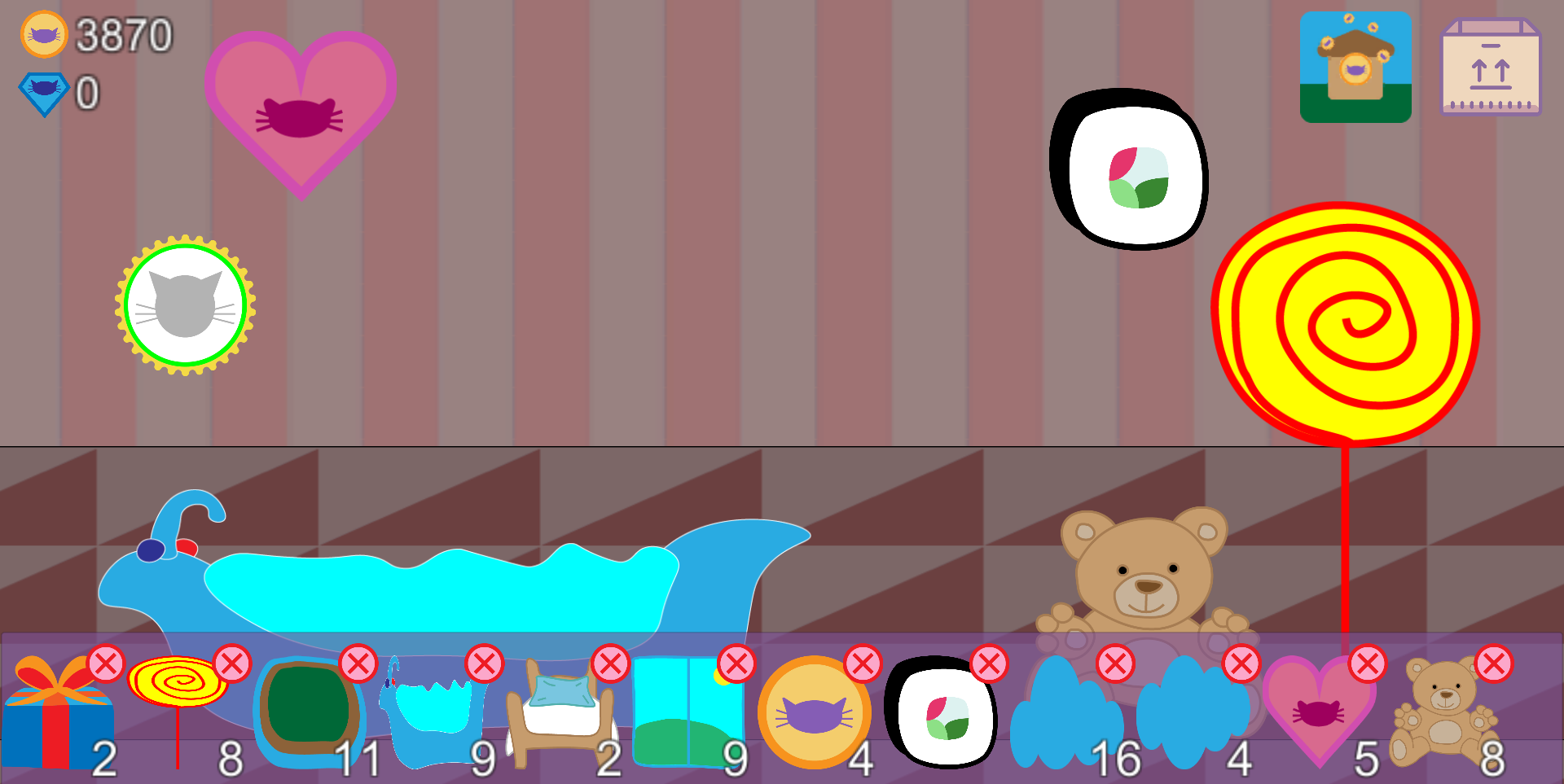


Рис. 3.4. Предмет вернулся в инвентарь

Тест №3. Нажимаем на крестик над предметом в инвентаре. Открывается панель продажи предмета (рис. 3.5). При нажатии кнопки «Да» уменьшается количество предметов в инвентаре и пополняется валюта (рис. 3.6).

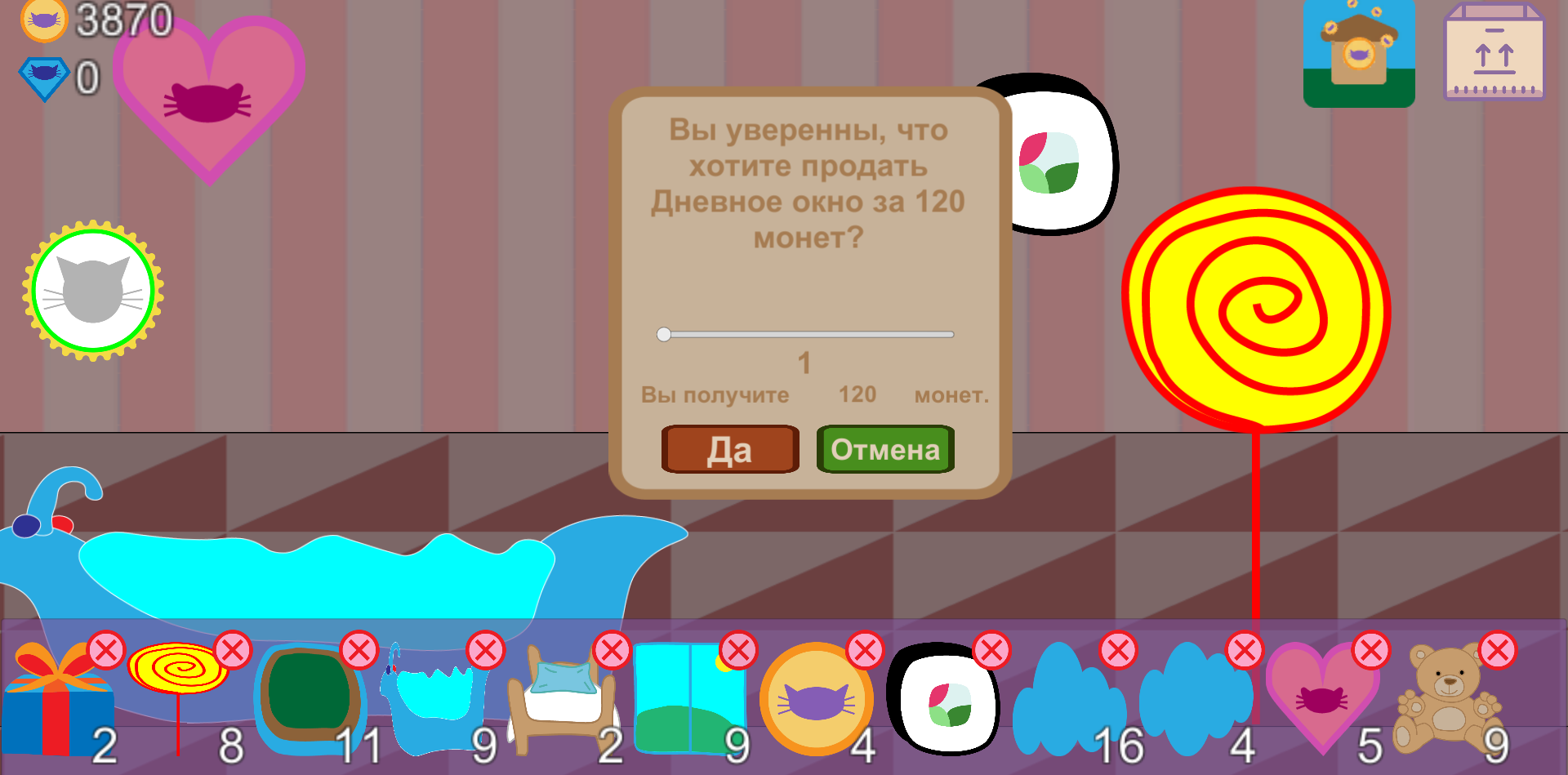


Рис. 3.5. Открытие панели продажи предмета

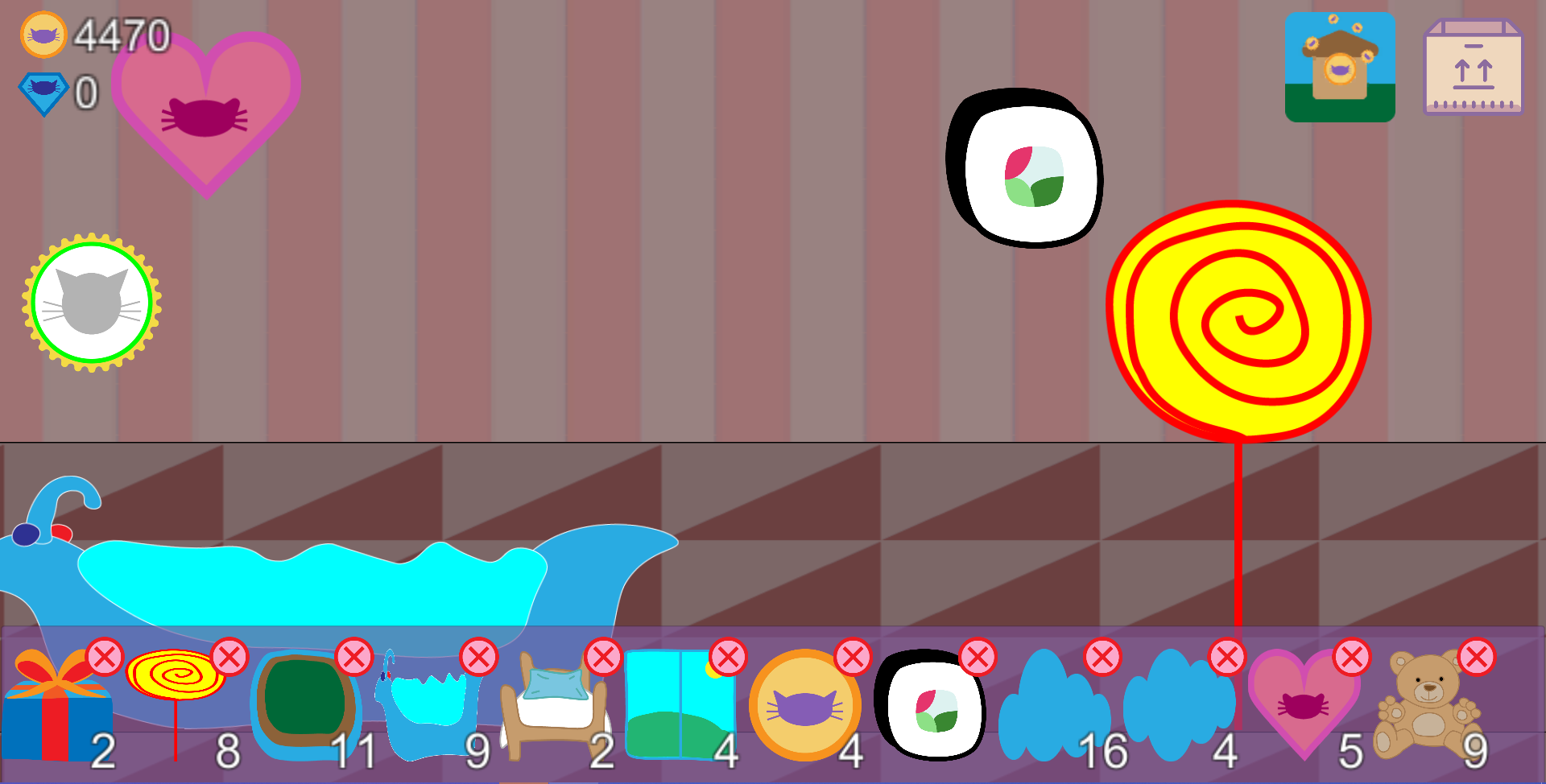


Рис. 3.6. Уменьшение количества предметов в инвентаре и пополнение валюты

Тест №4. Нажимаем на иконку улицы и попадаем на сцену улицы (рис. 3.7).



Рис. 3.7. Сцена улицы

Тест №5. Нажимаем на здание магазина и попадаем на сцену магазина (рис. 3.8).



Рис. 3.8. Сцена магазина

Тест №6. Нажимаем на предмет в магазине. Открывается панель покупки предмета (рис. 3.9). При нажатии кнопки «Да» уменьшается валюта (рис. 3.10).



Рис. 3.9. Открытие панели покупки предмета

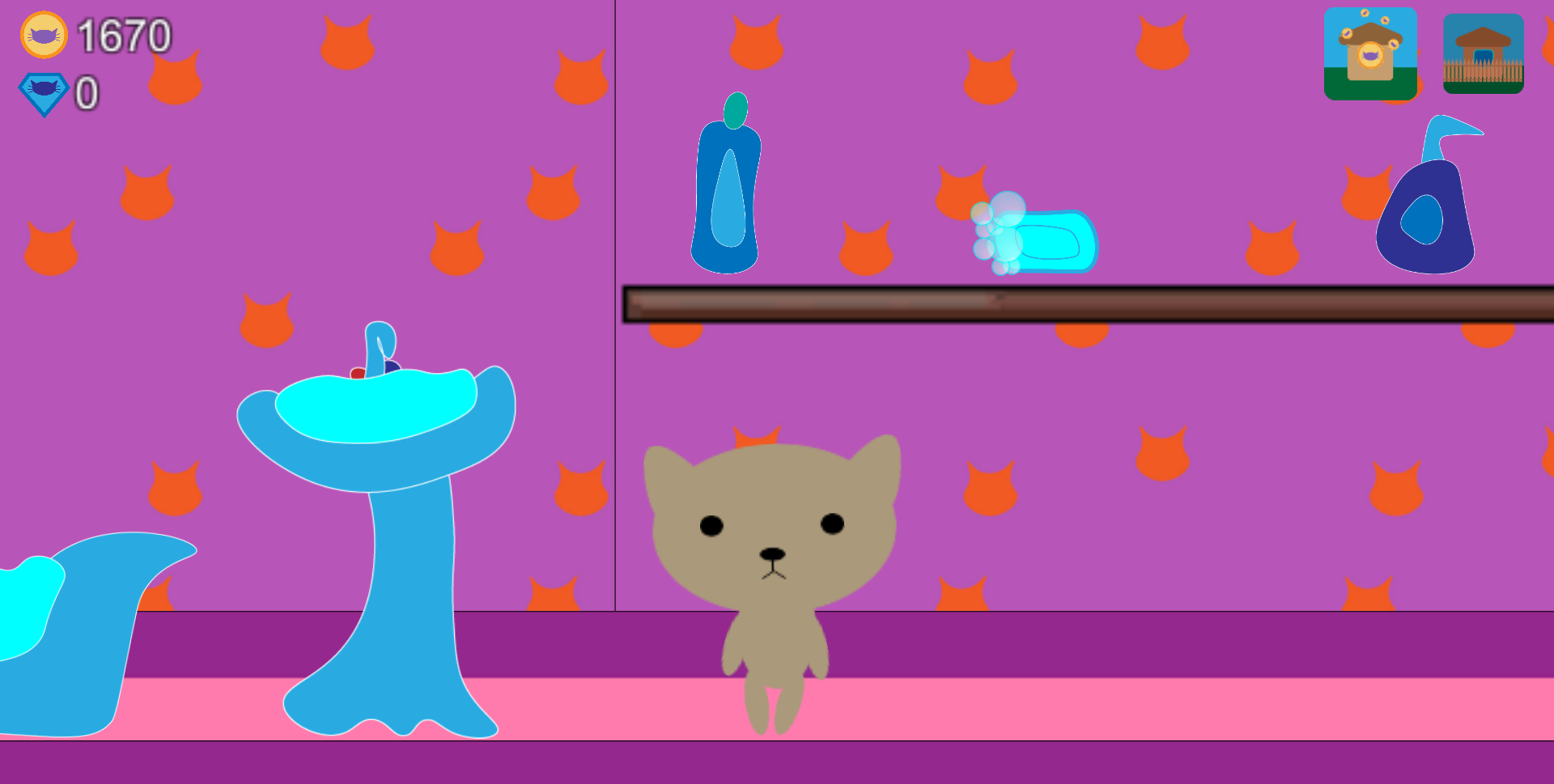


Рис. 3.10. Уменьшение валюты

Тест №7. Нажимаем на предмет в магазине. Открывается панель покупки предмета. Увеличиваем количество до суммы больше, чем имеется (рис. 3.11). При нажатии кнопки «Да» панель закрывается, валюта не меняется (рис. 3.12).



Рис. 3.11. Увеличение количества



Рис. 3.12. Валюта не меняется

Тест №8. Нажимаем кнопку «Play» на сцене мини-игры (рис. 3.13). На экране появляются иконки ошибок и начинают выскакивать котики из отверстий (рис. 3.14).

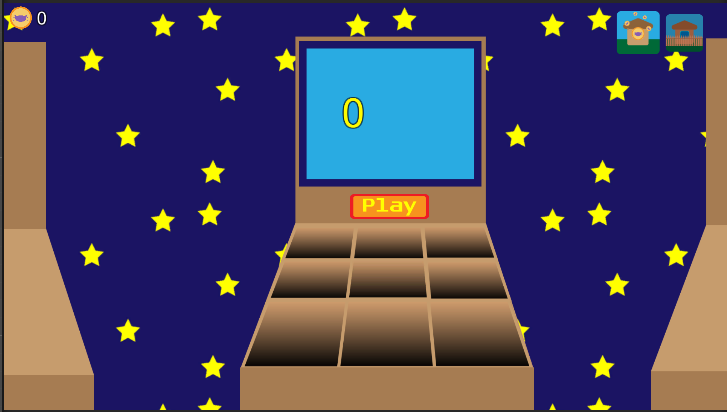


Рис. 3.13. Сцена мини-игры



Рис. 3.14. Начало игрового процесса

Тест №9. При нажатии на выскочившего котика экране увеличиваются очки (рис 3.15).



Рис. 3.15. Увеличение очков

Тест №10. При пропуске пяти выскочивших котиков игра заканчивается, очки начисляются (рис. 3.16).



Рис. 3.16. Конец игры

# 4 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

При запуске игры персонаж пользователя появляется в пустом игровом доме, пустым инвентарем и имеет некоторое количество игровой валюты (рис. 4.1).

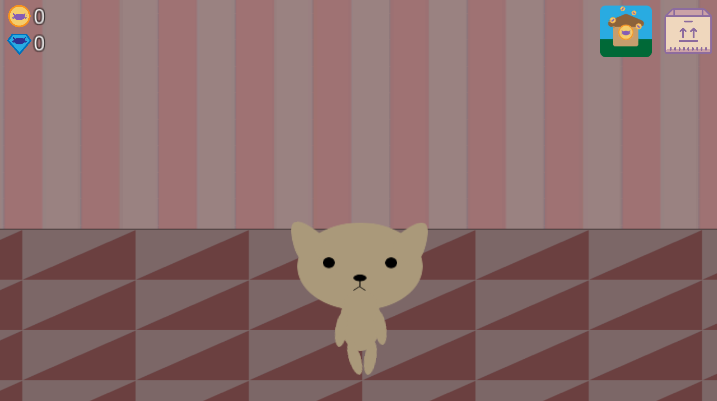


Рис. 4.1. Первое появление пользователя в игре

Пользователь имеет возможность посетить магазины и мини-игру, выйдя на улицу. Чтобы выйти на улицу необходимо нажать на иконку, на которой изображена улица (рис. 4.2).



Рис. 4.2. Иконка, на которой изображена улица

Чтобы попасть в определенный магазин, необходимо, находясь на улице, нажать на нужное здание, после чего пользователь попадет в магазин (рис. 4.3).



Рис. 4.3. Магазин

Чтобы приобрести вещь в магазине нужно нажать на неё, выбрать необходимое количество и нажать «Да». Если у пользователя хватает валюты на приобретение вещи, они появятся в его инвентаре (рис. 4.4).



Рис. 4.4. Покупка предмета в магазине

Чтобы открыть инвентарь, необходимо, находясь в игровом доме, нажать на иконку коробки (рис. 4.5).



Рис. 4.5. Иконка коробки

Для того, чтобы разместить предметы в игровом доме, необходимо нажать ПКМ на предмет в инвентаре, перетащить предмет на нужное место, и нажать на ЛКМ (рис. 4.6).



Рис. 4.6. Перетаскивание предмета в игровую комнату

Для того, чтобы вернуть предмет обратно в инвентарь, необходимо, в режиме инвентаря, нажать на предмет в комнате

Чтобы продать предмет, который находится в инвентаре, необходимо нажать на иконку крестика около нужного предмета. В появившемся окошке выбрать необходимое количество и нажать «Да» (рис. 4.7).

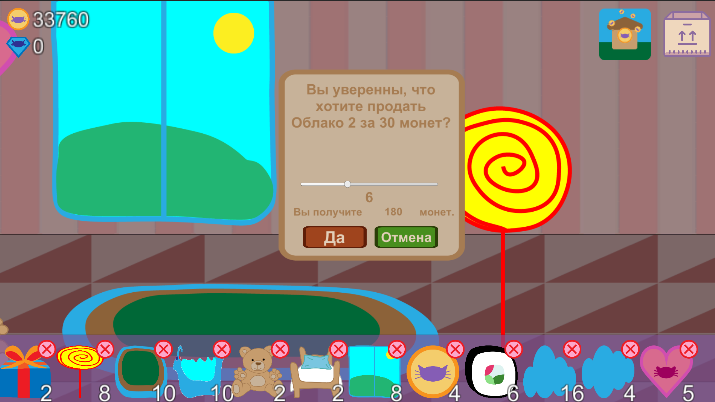


Рис. 4.6. Продажа предмета

# ЗАКЛЮЧЕНИЕ

В ходе курсового проекта была разработана игра «Shop Town». В приложении пользователь имеет возможность заработать игровую валюту, играя в интерактивную мини-игру, чтобы потратить её на предметы из магазина. Игрок может расставить купленные предметы в игровом доме по своему усмотрению.

В ходе курсового проекта был использован язык программирования С#, программы Visual Studio и Unity.

Программное средство было протестировано. Тестирование прошло успешно. В дальнейшем может быть модифицировано следующим образом: добавление новых магазинов и новых предметов, бонусы за ежедневный вход в игру, возможность изменять внешний вид персонажа, добавление новых мини-игр для зарабатывания игровой валюты, возможность существования нескольких комнат в доме.

Были выполнены цели курсового проекта: разработка программного средства для занимательного времяпровождения и отдыха, а также расширение, углубление знаний по дисциплине «конструирование программ и языки программирования». Получены навыки проектирования и создания программного обеспечения.

Были выполнены задачи курсового проекта:

- систематизация полученных знаний;

- углубление уровня и расширение объема профессионально значимых знаний, умений и навыков;

- формирование умений и навыков самостоятельной работы;

- практическое использование сериализации для хранения информации;

- разработка интуитивно простого интерфейса для пользователя.

# ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЯ:

1. Документация Unity [Электронный ресурс]. Режим доступа: https://docs.unity3d.com/Manual/index.html. Дата доступа: 03.03.2021.
2. Создание игры на Unity [Электронный ресурс]. Режим доступа: https://www.youtube.com/watch?v=u9RIoJNKERQ. Дата доступа: 08.03.2021.
3. Статьи о Universal Render Pipeline [Электронный ресурс]. Режим доступа:https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@10.5/manual/index.html. Дата доступа: 15.03.2021.
4. Основные настройки URP [Электронный ресурс]. Режим доступа: https://habr.com/ru/sandbox/141328/. Дата доступа: 20.03.2021.
5. Урок 2D UFO Unity [Электронный ресурс]. Режим доступа: https://learn.unity.com/project/2d-ufo-tutorial. Дата доступа: 25.03.2021.
6. Урок 2D Rogue Like Unity [Электронный ресурс]. Режим доступа: https://learn.unity.com/project/2d-roguelike-tutorial. Дата доступа: 27.03.2021.
7. Урок development techniques [Электронный ресурс]. Режим доступа: https://learn.unity.com/tutorial/mobile-development-techniques?language=en . Дата доступа 02.04.2021.

# ПРИЛОЖЕНИЕ 1 – ДИАГРАММА «СУЩНОСТЬ-СВЯЗЬ»

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
|  |  |  |  |  | *МКП.П1807-15 ПЛ* | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  | *Диаграмма «сущность-связь»* | *Лит.* | | | *Масса* | | *Масштаб* |
| *Изм* | *Лист* | *№ докум.* | *Подпись* | *Дата* |  | *у* |  |  | | *1:1* |
| *Разраб.* | | *Татур* |  |  |
| *Провер.* | | *Науменко* |  |  |
| *Т.Контр.* | |  |  |  | *Лист 1* | | | | *Листов 1* | |
| *Реценз.* | |  |  |  | *плакат* | *МКП* | | | | | |
| *Н.Контр.* | |  |  |  |
| *Утверд.* | |  |  |  |

# ПРИЛОЖЕНИЕ 2 – ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
|  |  |  |  |  | *МКП.П1807-15 ПЛ* | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  | *Диаграмма вариантов использования* | *Лит.* | | | *Масса* | | *Масштаб* |
| *Изм* | *Лист* | *№ докум.* | *Подпись* | *Дата* |  | *у* |  |  | | *1:1* |
| *Разраб.* | | *Татур* |  |  |
| *Провер.* | | *Науменко* |  |  |
| *Т.Контр.* | |  |  |  | *Лист 1* | | | | *Листов 1* | |
| *Реценз.* | |  |  |  | *плакат* | *МКП* | | | | | |
| *Н.Контр.* | |  |  |  |
| *Утверд.* | |  |  |  |

# ПРИЛОЖЕНИЕ 3 – ДИАГРАММА КЛАССОВ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
|  |  |  |  |  | *МКП.П1807-15 ПЛ* | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  | *Диаграмма классов* | *Лит.* | | | *Масса* | | *Масштаб* |
| *Изм* | *Лист* | *№ докум.* | *Подпись* | *Дата* |  | *у* |  |  | | *1:1* |
| *Разраб.* | | *Татур* |  |  |
| *Провер.* | | *Науменко* |  |  |
| *Т.Контр.* | |  |  |  | *Лист 1* | | | | *Листов 1* | |
| *Реценз.* | |  |  |  | *плакат* | *МКП* | | | | | |
| *Н.Контр.* | |  |  |  |
| *Утверд.* | |  |  |  |

ПРИЛОЖЕНИЕ 4 (СПРАВОЧНОЕ) – ЛИСТИНГ ПРОГРАММЫ

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

public class tab : MonoBehaviour

{

public EventSystem es;

public GameObject inv\_tab;

public GameObject cat;

bool isOpen;

void Start()

{

isOpen = inv\_tab.activeSelf;

GetComponent<Button>().onClick.AddListener(delegate

{

OnMouseDown();

});

CheckCat();

}

void Update()

{

}

private void OnMouseDown()

{

print("box");

isOpen = !isOpen;

inv\_tab.SetActive(isOpen);

CheckCat();

}

private void CheckCat()

{

if (isOpen)

{

cat.SetActive(false);

}

else

{

cat.SetActive(true);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class streetButton : MonoBehaviour

{

void Start()

{

GetComponent<Button>().onClick.AddListener(delegate

{

OnMouseDown();

});

}

void Update()

{

}

private void OnMouseDown()

{

print("shop");

SceneManager.LoadScene("StreetScene");

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class SkriptPlayer : MonoBehaviour

{

public float speed;

private Rigidbody2D rb;

private Vector2 moveVelosity;

private bool isFliped = false, flipState = false;

private SpriteRenderer sr;

Animator anim;

public Camera camera;

void Start()

{

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

sr = GetComponent<SpriteRenderer>();

if (SceneManager.GetActiveScene().name == "GameScene")

{

anim.SetBool("IsGameScene", true);

}

}

void Update()

{

if (SceneManager.GetActiveScene().name != "GameScene")

{

if (Input.GetMouseButton(0))

{

if (camera.transform.position.x+1>0)

{

Vector2 mouse = Camera.main.ScreenToWorldPoint(Input.mousePosition);

Flip(mouse);

transform.position = Vector2.MoveTowards(transform.position, new Vector2(mouse.x, transform.position.y), speed \* Time.deltaTime);

anim.SetBool("IsWalking", true);

camera.transform.position = new Vector3(transform.position.x, camera.transform.position.y,-10);

}

else

{

transform.position = new Vector2(0, transform.position.y);

camera.transform.position = new Vector3(0, camera.transform.position.y, camera.transform.position.z);

}

}

else

{

anim.SetBool("IsWalking", false);

}

}

}

private void Flip(Vector2 mouse)

{

if (mouse.x < transform.position.x)

{

isFliped = true;

}

else

{

isFliped = false;

}

if (isFliped)

{

if (!flipState)

{

transform.localScale \*= new Vector2(-1, 1);

flipState = true;

}

}

else

{

if (flipState)

{

transform.localScale \*= new Vector2(-1, 1);

flipState = false;

}

}

}

void FixedUpdate()

{

}

private void OnTriggerEnter2D(Collider2D collision)

{

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MoveIntoShop : MonoBehaviour

{

void Start()

{

}

void Update()

{

}

private void OnMouseDown()

{

SceneManager.LoadScene(name);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

using UnityEngine.UI;

public class ItemSkripts : MonoBehaviour

{

public int currID = -1;

public EventSystem es;

public GameObject movingObject;

public database data;

public Camera cam;

void Start()

{

GetComponent<Button>().onClick.AddListener(delegate { SelectObject(); });

}

void Update()

{

if (currID!=-1)

{

MoveObject();

}

}

public void SelectObject()

{

if (currID == -1)

{

currID = int.Parse(es.currentSelectedGameObject.name);

movingObject.GetComponent<SpriteRenderer>().sprite = data.items[currID].model.GetComponent<SpriteRenderer>().sprite;

movingObject.GetComponent<Transform>().localScale = data.items[currID].model.GetComponent<Transform>().localScale;

movingObject.gameObject.SetActive(true);

Destroy(es.currentSelectedGameObject);

print(es.currentSelectedGameObject.name);

}

else

{

currID = -1;

movingObject.gameObject.SetActive(false);

}

}

public void MoveObject()

{

Vector3 pos = Input.mousePosition ;

pos.z = 1

movingObject.GetComponent<Transform>().position = cam.ScreenToWorldPoint(pos);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class homeButton : MonoBehaviour

{

void Start()

{

GetComponent<Button>().onClick.AddListener(delegate

{

OnMouseDown();

});

}

void Update()

{

}

private void OnMouseDown()

{

print("home");

SceneManager.LoadScene("HouseScene");

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.Threading.Tasks;

using System.IO;

public class GameScript : MonoBehaviour

{

public SavingValues savingValues;

public GameObject playButton;

public GameObject misses;

public GameObject points;

public GameObject gameOver;

public GameObject coins;

public Camera cam;

public Sprite cat;

Animator anim;

public Sprite bonkedCat;

public GameObject movingObject;

private float currentSpeed = 1000;

public GameObject[,] catsGameObjs = new GameObject[3, 3];

public bool[,] catsGameObjsIsActive = new bool[3, 3];

private bool IsGameGoing = false;

private bool IsGameStarted = false;

private int money = 0;

private int missCount = 5;

void Start()

{

if (File.Exists(Application.dataPath + "/save.gamesave"))

{

string jsonText = File.ReadAllText(Application.dataPath + "/save.gamesave");

savingValues = JsonUtility.FromJson<SavingValues>(jsonText);

print($"{Application.dataPath + "/save.gamesave"} exist");

}

else

{

savingValues = new SavingValues(0, 0, new List<FixedObjects>(), new List<ItemIdCount>());

File.WriteAllText(Application.dataPath + "/save.gamesave", JsonUtility.ToJson(savingValues));

}

coins.GetComponent<Text>().text = savingValues.coins.ToString();

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

catsGameObjs[i, j] = GameObject.Find($"{i}{j}").transform.GetChild(0).gameObject;

catsGameObjsIsActive[i, j] = false;

catsGameObjs[i, j].SetActive(false);

}

}

playButton.GetComponent<Button>().onClick.AddListener(delegate { StartGame(); });

}

public void WasBonked()

{

print("ouch");

points.GetComponent<Text>().text = (int.Parse(points.GetComponent<Text>().text) + 10).ToString();

}

private void StartGame()

{

if (!IsGameStarted)

{

gameOver.gameObject.SetActive(false);

IsGameStarted = true;

print("Game is started!");

points.GetComponent<Text>().text = "0";

missCount = 5;

for (int i = 0; i < misses.transform.childCount; i++)

{

misses.transform.GetChild(i).gameObject.SetActive(true);

}

movingObject.gameObject.SetActive(true);

IsGameGoing = true;

}

}

private async void GameStep()

{

IsGameGoing = false;

bool IsFreePlase = true;

int i = -1;

int j = -1;

while (IsFreePlase)

{

i = UnityEngine.Random.Range(0, 3);

j = UnityEngine.Random.Range(0, 3);

if (!catsGameObjsIsActive[i, j])

{

IsFreePlase = false;

catsGameObjsIsActive[i, j] = true;

catsGameObjs[i, j].SetActive(true);

}

}

await Task.Delay(TimeSpan.FromMilliseconds(currentSpeed));

IsGameGoing = true;

catsGameObjsIsActive[i, j] = false;

if (catsGameObjs[i, j].GetComponent<SpriteRenderer>().sprite == cat)

{

print("miss");

if (missCount != 1)

{

missCount--;

misses.transform.GetChild(missCount).gameObject.SetActive(false);

}

else

{

missCount--;

misses.transform.GetChild(missCount).gameObject.SetActive(false);

GameOver();

}

}

catsGameObjs[i, j].GetComponent<SpriteRenderer>().sprite = cat;

catsGameObjs[i, j].SetActive(false);

}

private void GameOver()

{

IsGameGoing = false;

IsGameStarted = false;

gameOver.gameObject.SetActive(true);

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

catsGameObjsIsActive[i, j] = false;

catsGameObjs[i, j].SetActive(false);

}

}

string jsonText = File.ReadAllText(Application.dataPath + "/save.gamesave");

savingValues = JsonUtility.FromJson<SavingValues>(jsonText);

print($"{Application.dataPath + "/save.gamesave"} exist");

savingValues.coins += int.Parse(points.GetComponent<Text>().text);

File.WriteAllText(Application.dataPath + "/save.gamesave", JsonUtility.ToJson(savingValues));

coins.GetComponent<Text>().text = savingValues.coins.ToString();

}

void Update()

{

Vector2 pos = Input.mousePosition;

pos = cam.ScreenToWorldPoint(pos);

movingObject.transform.position = new Vector3(pos.x, pos.y, -8);

movingObject.transform.localScale = new Vector3(0.55f, 0.55f, 0.55f);

print(misses.transform.childCount);

if (Input.GetMouseButtonDown(0))

{

print("тап");

movingObject.transform.Rotate(0.0f, 0.0f, 45.0f);

}

if (Input.GetMouseButtonUp(0))

{

movingObject.transform.Rotate(0.0f, 0.0f, -45.0f);

}

if (IsGameGoing)

{

GameStep();

if (currentSpeed >= 400)

{

currentSpeed -= 8;

}

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ForCatsCollisions : MonoBehaviour

{

public Sprite bonk;

public Sprite cat;

void Start()

{

}

void Update()

{

}

private void OnMouseDown()

{

if (GetComponent<SpriteRenderer>().sprite != bonk)

{

GetComponent<SpriteRenderer>().sprite = bonk;

GameObject.Find("Main Camera").SendMessage("WasBonked");

}

}

private void OnMouseExit()

{

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class database : MonoBehaviour

{

public List<Item> items = new List<Item>();

}

[System.Serializable]

public class Item

{

public int id;

public string name;

public Sprite image;

public GameObject model;

public int price;

public int sellPrice;

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

using System.IO;

using UnityEngine.SceneManagement;

using System;

public class inventory : MonoBehaviour

{

public database data;

public List<ItemInventory> items = new List<ItemInventory>();

public GameObject gameObjectShow;// (?)

public GameObject InventoryMainObject;

public int itemCount;// я изменила макс на колво простов инвентаре

public Camera cam;

public EventSystem es;

public int currID;

public ItemInventory currItem;

//public RectTransform movingObject;

public GameObject movingObject;

public Vector3 offset;

public SavingValues savingValues;

public GameObject sellingPanel;

private GameObject SellingItem;

public GameObject BuyingPanel;

private Item ByingItem;

public void Start()

{

Load();

if (SceneManager.GetActiveScene().name == "HouseScene")

{

sellingPanel.transform.GetChild(1).GetComponent<Button>().onClick.AddListener(delegate { YesSell(); });

sellingPanel.transform.GetChild(2).GetComponent<Button>().onClick.AddListener(delegate { CansellSell(); });

sellingPanel.transform.GetChild(3).GetComponent<Slider>().onValueChanged.AddListener(delegate { ChangeAmount(); });

}

if (SceneManager.GetActiveScene().name == "FurnitureShopScene" || SceneManager.GetActiveScene().name == "SoapShop" || SceneManager.GetActiveScene().name == "StickerShop")

{

BuyingPanel.transform.GetChild(1).GetComponent<Button>().onClick.AddListener(delegate { YesSellShop(); });

BuyingPanel.transform.GetChild(2).GetComponent<Button>().onClick.AddListener(delegate { CansellSellShop(); });

BuyingPanel.transform.GetChild(3).GetComponent<Slider>().onValueChanged.AddListener(delegate { ChangeAmountShop(); });

}

}

private void ChangeAmountShop()

{

BuyingPanel.transform.GetChild(3).GetComponent<Slider>().transform.GetChild(0).GetComponent<Text>().text = BuyingPanel.transform.GetChild(3).GetComponent<Slider>().value.ToString();

BuyingPanel.transform.GetChild(5).GetComponent<Text>().text = (ByingItem.price \* BuyingPanel.transform.GetChild(3).GetComponent<Slider>().value).ToString();

}

private void CansellSellShop()

{

BuyingPanel.transform.GetChild(0).GetComponent<Text>().text = "Вы уверенны, что хотите купить itemName за amount монет?";

BuyingPanel.transform.GetChild(3).GetComponent<Slider>().value = 1;

BuyingPanel.SetActive(false);

}

private void YesSellShop()

{

if (int.Parse(GameObject.Find("coinAmount").GetComponent<Text>().text) >= int.Parse(BuyingPanel.transform.GetChild(5).GetComponent<Text>().text))

{

GameObject.Find("coinAmount").GetComponent<Text>().text = (int.Parse(GameObject.Find("coinAmount").GetComponent<Text>().text) - int.Parse(BuyingPanel.transform.GetChild(5).GetComponent<Text>().text)).ToString();

savingValues.coins = int.Parse(GameObject.Find("coinAmount").GetComponent<Text>().text);

int countt = 0;

for (int i = 0; i < savingValues.items.Count; i++)

{

if (savingValues.items[i].id == ByingItem.id)

{

countt = savingValues.items[i].count;

savingValues.items.Remove(savingValues.items[i]);

}

}

savingValues.items.Add(new ItemIdCount(ByingItem.id, countt+int.Parse(BuyingPanel.transform.GetChild(3).GetComponent<Slider>().value.ToString())));

File.WriteAllText(Application.dataPath + "/save.gamesave", JsonUtility.ToJson(savingValues));

print("saved");

CansellSellShop();

}

CansellSellShop();

}

private void ClickOnItemInShop(int id)

{

print($"ClickOnItemInShop and it is {data.items[id].name}");

ByingItem = data.items[id];

BuyingPanel.transform.GetChild(5).GetComponent<Text>().text = ByingItem.price.ToString();

BuyingPanel.transform.GetChild(0).GetComponent<Text>().text = BuyingPanel.transform.GetChild(0).GetComponent<Text>().text.Replace("itemName",ByingItem.name

BuyingPanel.transform.GetChild(0).GetComponent<Text>().text = BuyingPanel.transform.GetChild(0).GetComponent<Text>().text.Replace("amount", ByingItem.price.ToString());

BuyingPanel.SetActive(true);

}

private void ChangeAmount()

{

sellingPanel.transform.GetChild(3).GetComponent<Slider>().transform.GetChild(0).GetComponent<Text>().text = sellingPanel.transform.GetChild(3).GetComponent<Slider>().value.ToString();

if (SellingItem != null)

{

sellingPanel.transform.GetChild(5).GetComponent<Text>().text = (data.items[items[ int.Parse(SellingItem.name)].id].sellPrice \* sellingPanel.transform.GetChild(3).GetComponent<Slider>().value).ToString();

}

}

private void CansellSell()

{

sellingPanel.transform.GetChild(0).GetComponent<Text>().text = "Вы уверенны, что хотите продать itemName за amount монет?";

sellingPanel.transform.GetChild(3).GetComponent<Slider>().value = 1;

sellingPanel.SetActive(false);

}

private void YesSell()

{

for (int j = 0; j < sellingPanel.transform.GetChild(3).GetComponent<Slider>().value; j++)

{

if (SellingItem.transform.GetChild(0).GetComponentInChildren<Text>().text == "")

{

print(items.Remove(items[int.Parse(SellingItem.name)]));

for (int i = int.Parse(SellingItem.name); i < items.Count; i++)

{

items[i].gameObject.name = (int.Parse(items[i].gameObject.name) - 1).ToString();

}

Destroy(SellingItem);

}

else

{

items[int.Parse(SellingItem.name)].count--;

if (SellingItem.transform.GetChild(0).GetComponentInChildren<Text>().text == "2")

{

SellingItem.transform.GetChild(0).GetComponentInChildren<Text>().text = "";

}

else

{

SellingItem.transform.GetChild(0).GetComponentInChildren<Text>().text = (int.Parse(SellingItem.transform.GetChild(0).GetComponentInChildren<Text>().text) - 1).ToString();

}

}

}

GameObject.Find("coinAmount").GetComponent<Text>().text = (int.Parse(GameObject.Find("coinAmount").GetComponent<Text>().text) + int.Parse(sellingPanel.transform.GetChild(5).GetComponent<Text>().text)).ToString();

Save();

CansellSell();

}

public void Update()

{

if (currID!=-1)

{

MoveObject();

}

if (Input.GetMouseButtonDown(1) && currID!=-1)

{

InstnantiateGameObj();

}

//UpdateInventory();

}

public void InsertIntoInv(int id, Item item, int count)

{

}

public void AddItem(int id, Item item, int count)

{

items[id].id = item.id;

items[id].count = count;

items[id].gameObject.GetComponentInChildren<Image>().sprite = item.image;

if (count > 1 && item.id != 0)// почему нуль??

{

items[id].gameObject.GetComponentInChildren<Text>().text = count.ToString();

}

else

{

items[id].gameObject.GetComponentInChildren<Text>().text = "";

}

}

public void AddInventoryItem(int id, ItemInventory invItem)

{

items[id].id = invItem.id;

items[id].count = invItem.count;

items[id].gameObject.GetComponentInChildren<Image>().sprite = data.items[invItem.id].image;

if (invItem.count > 1 && invItem.id != 0)// почему нуль??

{

items[id].gameObject.GetComponentInChildren<Text>().text = invItem.count.ToString();

}

else

{

items[id].gameObject.GetComponentInChildren<Text>().text = "";

}

}

public void AddGraphics()

{

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

{

GameObject newItem = Instantiate(gameObjectShow, InventoryMainObject.transform) as GameObject;

newItem.name = i.ToString();

ItemInventory ii = new ItemInventory();

ii.gameObject = newItem;

RectTransform rt = newItem.GetComponent<RectTransform>();

rt.localPosition = new Vector3(0, 0, 0);

rt.localScale = new Vector3(1, 1, 1);

newItem.GetComponent<RectTransform>().localScale = new Vector3(1, 1, 1);

Button itemButton = newItem.GetComponent<Button>();

newItem.transform.GetChild(1).GetComponent<Button>().onClick.AddListener(delegate { SellItemPanel(); });

itemButton.onClick.AddListener(delegate

{

SelectObject();

});

items.Add(ii);

}

}

public void AddGraphicsI(int i)

{

int isItemExist = -1;

foreach (var item in items)

{

if (item.id==i)

{

isItemExist = int.Parse(item.gameObject.name);

}

}

if (isItemExist != -1)

{

items[isItemExist].count++;

UpdateInventory();

}

else

{

GameObject newItem = Instantiate(gameObjectShow, InventoryMainObject.transform) as GameObject;

newItem.name = items.Count.ToString();

ItemInventory ii = new ItemInventory();

ii.gameObject = newItem;

RectTransform rt = newItem.GetComponent<RectTransform>();

rt.localPosition = new Vector3(0, 0, 0);

rt.localScale = new Vector3(1, 1, 1);

newItem.GetComponent<RectTransform>().localScale = new Vector3(1, 1, 1);

Button itemButton = newItem.GetComponent<Button>();

itemButton.onClick.AddListener(delegate

{

SelectObject();

});

newItem.transform.GetChild(1).GetComponent<Button>().onClick.AddListener(delegate { SellItemPanel(); });

items.Add(ii);

AddItem(items.Count-1, data.items[i],1);

UpdateInventory();

}

}

public void RemoveFromInv(int InvIndx)

{

items.Remove(items[InvIndx]);

Destroy(es.currentSelectedGameObject);

}

public void SelectObject()

{

if (currID == -1)

{

currID = int.Parse(es.currentSelectedGameObject.name);

currItem = CopyInventoryItem(items[currID]);

print(es.currentSelectedGameObject.GetType());

data.items[currItem.id].model.GetComponent<SpriteRenderer>().sprite;

movingObject.GetComponent<SpriteRenderer>().sprite = data.items[currItem.id].model.GetComponent<SpriteRenderer>().sprite;

cam.WorldToScreenPoint(data.items[currItem.id].model.GetComponent<SpriteRenderer>().size);

print(data.items[currItem.id].name);

movingObject.GetComponent<Transform>().localScale = data.items[currItem.id].model.GetComponent<Transform>().localScale;

movingObject.gameObject.SetActive(true);

if (es.currentSelectedGameObject.GetComponentInChildren<Text>().text=="")

{

print(items.Remove(items[currID]));

for (int i = currID; i < items.Count; i++)

{

items[i].gameObject.name = (int.Parse(items[i].gameObject.name)-1).ToString();

}

Destroy(es.currentSelectedGameObject);

}

else

{

items[currID].count--;

if (es.currentSelectedGameObject.GetComponentInChildren<Text>().text == "2")

{

es.currentSelectedGameObject.GetComponentInChildren<Text>().text = "";

}

else

{

es.currentSelectedGameObject.GetComponentInChildren<Text>().text = (int.Parse(es.currentSelectedGameObject.GetComponentInChildren<Text>().text)-1).ToString();

}

}

}

else

{

AddInventoryItem(currID, items[int.Parse(es.currentSelectedGameObject.name)]);//при перетаскивании обьекта типо заменяется плейс на ноль

AddInventoryItem(int.Parse(es.currentSelectedGameObject.name), currItem);

currID = -1;

movingObject.gameObject.SetActive(false);

}

}

public void InstnantiateGameObj()

{

var GameObj =(GameObject) Instantiate(data.items[currItem.id].model.gameObject.gameObject, cam.ScreenToWorldPoint(Input.mousePosition)+new Vector3(0,0,80),Quaternion.identity);

GameObj.name = currItem.id.ToString();

GameObj.GetComponent<Button>().onClick.AddListener(delegate { SelectItem(); });

currID = -1;

movingObject.gameObject.SetActive(false);

}

private void SellItemPanel()

{

SellingItem = es.currentSelectedGameObject.transform.parent.gameObject;

if (SellingItem.transform.GetChild(0).GetComponent<Text>().text!="")

{

sellingPanel.transform.GetChild(3).GetComponent<Slider>().maxValue = int.Parse(SellingItem.transform.GetChild(0).GetComponent<Text>().text);

}

else

{

sellingPanel.transform.GetChild(3).GetComponent<Slider>().maxValue = 1;

}

sellingPanel.transform.GetChild(0).GetComponent<Text>().text = sellingPanel.transform.GetChild(0).GetComponent<Text>().text.Replace("itemName", data.items[items[int.Parse(es.currentSelectedGameObject.transform.parent.name)].id].name);

sellingPanel.transform.GetChild(0).GetComponent<Text>().text = sellingPanel.transform.GetChild(0).GetComponent<Text>().text.Replace("amount", data.items[items[int.Parse(es.currentSelectedGameObject.transform.parent.name)].id].sellPrice.ToString());

sellingPanel.transform.GetChild(5).GetComponent<Text>().text = data.items[items[int.Parse(es.currentSelectedGameObject.transform.parent.name)].id].sellPrice.ToString();

sellingPanel.SetActive(true);

}

private void SelectItem()

{

if (currID == -1)

{

//print($"{es.currentSelectedGameObject.name} was pressed!!!!!!!!!!!!!!!!!!!!!");

currID = int.Parse(es.currentSelectedGameObject.name);

movingObject.GetComponent<SpriteRenderer>().sprite = data.items[currID].model.GetComponent<SpriteRenderer>().sprite;

movingObject.GetComponent<Transform>().localScale = data.items[currID].model.GetComponent<Transform>().localScale;

movingObject.gameObject.SetActive(true);

Destroy(es.currentSelectedGameObject);

}

else

{

print($"{es.currentSelectedGameObject.name} - меня нажалиии!");

currID = -1;

movingObject.gameObject.SetActive(false);

}

}

public void MoveObject()

{

Vector3 pos = Input.mousePosition + offset;

pos.z = InventoryMainObject.GetComponent<RectTransform>().position.z;

movingObject.GetComponent<Transform>().position = cam.ScreenToWorldPoint(pos);

}

public void UpdateInventory()

{

for (int i = 0; i < items.Count; i++)

{

if (items[i].id !=0 && items[i].count>1)

{

items[i].gameObject.GetComponentInChildren<Text>().text = items[i].count.ToString();

}

else

{

items[i].gameObject.GetComponentInChildren<Text>().text = "";

}

items[i].gameObject.GetComponentInChildren<Image>().sprite = data.items[items[i].id].image;

}

}

public void ItemGoingBack(string values)

{

var strs = values.Split();

AddGraphicsI(int.Parse(strs[3]));

savingValues.fixedObjects.Remove(new FixedObjects(float.Parse(strs[0]), float.Parse(strs[1]), float.Parse(strs[2]), int.Parse(strs[3])));

Save();

}

public ItemInventory CopyInventoryItem(ItemInventory old)

{

ItemInventory New = new ItemInventory();

New.id = old.id;

New.gameObject = old.gameObject;

New.count = old.count;

return New;

}

public void FixObj(string values)

{

var strs = values.Split();

if (!savingValues.fixedObjects.Contains(new FixedObjects(float.Parse(strs[0]), float.Parse(strs[1]), float.Parse(strs[2]), int.Parse(strs[3]))))

{

savingValues.fixedObjects.Add(new FixedObjects(float.Parse(strs[0]), float.Parse(strs[1]), float.Parse(strs[2]),int.Parse(strs[3])));

}

print("fixed");

Save();

}

public void Save()

{

List<ItemIdCount> newList = new List<ItemIdCount>();

for (int i = 0; i < items.Count; i++)

{

newList.Add(new ItemIdCount(items[i].id,items[i].count));

}

savingValues.coins = int.Parse(GameObject.Find("coinAmount").GetComponent<Text>().text);

savingValues.diamonds = int.Parse(GameObject.Find("diamondAmount").GetComponent<Text>().text);

if (SceneManager.GetActiveScene().name == "HouseScene")

{

savingValues.items = newList;

}

File.WriteAllText(Application.dataPath + "/save.gamesave", JsonUtility.ToJson(savingValues));

print("saved");

}

public void Load()

{

if (File.Exists(Application.dataPath + "/save.gamesave"))

{

string jsonText = File.ReadAllText(Application.dataPath + "/save.gamesave");

savingValues = JsonUtility.FromJson<SavingValues>(jsonText);

print($"{Application.dataPath + "/save.gamesave"} exist");

}

else

{

savingValues = new SavingValues(0, 0, new List<FixedObjects>(), new List<ItemIdCount>());

Save();

}

GameObject.Find("coinAmount").GetComponent<Text>().text = savingValues.coins.ToString();

GameObject.Find("diamondAmount").GetComponent<Text>().text = savingValues.diamonds.ToString();

if (SceneManager.GetActiveScene().name == "HouseScene")

{

foreach (var obj in savingValues.fixedObjects)

{

var GameObj = (GameObject)Instantiate(data.items[obj.id].model.gameObject.gameObject, new Vector3(obj.x, obj.y, obj.z), Quaternion.identity);

GameObj.GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Static;

GameObj.name = obj.id.ToString();

}

int indForInv = 0;

foreach (var inv in savingValues.items)

{

GameObject newItem = Instantiate(gameObjectShow, InventoryMainObject.transform) as GameObject;

newItem.name = indForInv.ToString();

ItemInventory ii = new ItemInventory();

ii.gameObject = newItem;

RectTransform rt = newItem.GetComponent<RectTransform>();

rt.localPosition = new Vector3(0, 0, 0);

rt.localScale = new Vector3(1, 1, 1);

newItem.GetComponent<RectTransform>().localScale = new Vector3(1, 1, 1);

Button itemButton = newItem.GetComponent<Button>();

itemButton.onClick.AddListener(delegate

{

SelectObject();

});

newItem.transform.GetChild(1).GetComponent<Button>().onClick.AddListener(delegate { SellItemPanel(); });

items.Add(ii);

AddItem(indForInv, data.items[inv.id], inv.count);

indForInv++;

}

UpdateInventory();

}

}

}

[System.Serializable]

public struct SavingValues

{

public int coins;

public int diamonds;

public List<FixedObjects> fixedObjects;

public List<ItemIdCount> items;

public SavingValues(int coins, int diamonds, List<FixedObjects> fixedObjects, List<ItemIdCount> items)

{

this.coins = coins;

this.diamonds = diamonds;

this.fixedObjects = fixedObjects;

this.items = items;

}

}

[System.Serializable]

public struct ItemIdCount

{

public int id;

public int count;

public ItemIdCount(int id, int count)

{

this.id = id;

this.count = count;

}

}

[System.Serializable]

public struct FixedObjects

{

public float x,y,z;

public int id;

public FixedObjects(float x, float y, float z, int id)

{

this.x = x;

this.y = y;

this.z = z;

this.id = id;

}

}

[System.Serializable]

public class ItemInventory

{

public int id;

public GameObject gameObject;// (?)

public int count;

} using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using System.Threading.Tasks;

using System.IO;

using UnityEngine.UI;

using System;

public class skriptForDeleteObj : MonoBehaviour

{

void Start()

{

}

void Update()

{

if (SceneManager.GetActiveScene().name == "HouseScene")

{

if (Input.GetMouseButtonUp(1))

{

print("up");

SavingFixing();

}

}

}

private void OnMouseDown()

{

if (SceneManager.GetActiveScene().name == "HouseScene")

{

try

{

if (GameObject.Find("inventory\_tab").activeSelf)

{

Destroy(gameObject);

GameObject.Find("inventory\_tab").SendMessageUpwards("ItemGoingBack", $"{transform.position.x} {transform.position.y} {transform.position.z} {int.Parse(name)}");

}

}

catch (System.Exception)

{

}

}

if (SceneManager.GetActiveScene().name == "FurnitureShopScene" || SceneManager.GetActiveScene().name == "SoapShop" || SceneManager.GetActiveScene().name == "StickerShop")

{

GameObject.Find("Main Camera").SendMessage("ClickOnItemInShop", int.Parse(name));

}

}

private async void SavingFixing()

{

await Task.Delay(TimeSpan.FromMilliseconds(1000));

if (SceneManager.GetActiveScene().name == "HouseScene" || SceneManager.GetActiveScene().name == "SoapShop" || SceneManager.GetActiveScene().name == "StickerShop")

{

GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Static;

GameObject.Find("inventory\_tab").SendMessageUpwards("FixObj", $"{transform.position.x} {transform.position.y} {transform.position.z} {int.Parse(name)}");

print("sended");

}

}

[System.Serializable]

public struct FixedObjects

{

public float x, y, z;

public int id;

public FixedObjects(float x, float y, float z, int id)

{

this.x = x;

this.y = y;

this.z = z;

this.id = id;

}

}

}