# Руководство программиста

## Назначение и условия применения программы

Программа является игрой для ПК, средством развлечения и предназначена для позитивного времяпровождения. Суть игры заключается в том, что игрок должен управлять предоставляемым ему персонажем так, чтобы тот смог добраться до правого конца уровня и не задеть ловушки более двух раз.

Условия, необходимые для корректного запуска игры:

* + - * ОС Windows 7 или выше 64 бит;
      * .NET Framework 4.5.2.

Минимальные системные требования:

* + - * частота процессора не менее 1 ГГц;
      * операционная система Windows 7 или выше;
      * 1024 Мбайт оперативной памяти;
      * мышь, клавиатура;
      * 240 Мбайт свободного места на жестком диске;
      * разрешение экрана не менее 1280x720.

Рекомендуемые системные требования:

* + - * частота процессора не менее 1.7 ГГц;
      * операционная система Windows 7 или выше;
      * 3 Гб оперативной памяти;
      * мышь, клавиатура;
      * 380 Мбайт свободного места на жестком диске;
      * разрешение экрана не менее 1366x768.

Требования к персоналу (программисту):

Программист должен обладать практическими навыками работы с пользовательским интерфейсом среды разработки Unity, знаниями языка C# по отношению к используемым скриптам для создания игр, а также знать и уметь пользоваться системой контроля версий GitHub.

## Характеристика программы

Средняя длительность прохождения: 24 минуты.

Идеальная длительность прохождения: 7 минут.

Программа не сохраняет данные – при перезапуске все данные (достижения, настройки) возвращаются в исходное состояние.

Быстродействие программы не зависит от качества ПК (при соответствии системным требованиям), то есть при повышении мощности компьютера быстродействие не увеличивается – игра стабильно работает на любых сборках системы.

Программа является автономной и способна работать без подключения к сети Интернет.

Все средства контроля правильности выполнения программы заложены в используемых скриптах, написанных на C#, за счет проверок условий выполнения того или иного метода.

Система предоставляет возможность ручного контроля процесса – управление персонажем по нажатию определенных клавиш. Во время работы игры происходит регистрация полученных достижения в ходе работы текущего запуска. Также позволяет изменять предложенные настройки игры, просматривать список достижений и разработчиков.

## Обращение к программе

Запуск готовой игры происходит через файл в формате .exe, который находится в корневой папке игры Grumpy Tree Friends (Рисунок 1).

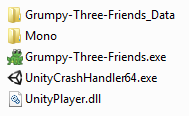


Рисунок 1 – Схема расположения файлов игры в проводнике

Весь исходный код программы расположен на удалённом репозитории GitHub по адресу: <https://github.com/SaPeUlDi/Grumpy-Three-Friends/>. Для внесения каких-либо изменений в программу, необходимо:

1. Войти в аккаунт GitHub (если его нет, необходимо пройти регистрацию и после войти).
2. Зайти на удалённый репозиторий проекта и создать fork. (как это сделать, можно узнать на этой странице: [https://git-scm.com/book/ru/v2/GitHub-Внесение-собственного-вклада-в-проекты](https://git-scm.com/book/ru/v2/GitHub-Внесение-собственного-вклада-в-проекты%20)).
3. Создать локальную копию репозитория.
4. Внести изменения.
5. Запушить на fork.
6. Создать запрос pull request'а в основной репозиторий ветик Unstable и ожидать одобрения запроса.
7. После того, как запрос будет одобрен, совершить pull request и merge в ветку Unstable.

Открыть сам проект в Unity можно, если зайти в программу Unity, нажать кнопку Open и выбрать корневую папку проекта. Структура папки разработки показана на Рисунке 2.

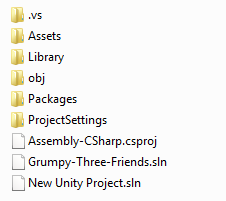


Рисунок 2 – Схема расположения файлов проекта в проводнике

При открытии проекта будет показан последний открытый разработчиком уровень при предыдущей работе с этим проектом в Unity.

## Входные и выходные данные

Программа может принимать данные от пользователя только для настройки громкости звука и переключения оконного\полноэкранного режима работы. Остальные данные задаются фиксировано в проекте Unity.

Реализация выходной информации не предусмотрена.

## Структура системы

Проект состоит из модулей-сцен, 6 из них объединены при помощи основного скрипта CharacterController.cs и могут независимо друг от друга исключаться из проекта для доработки безболезненно для всей системы при правильных изменениях названий сцен внутри скрипта в месте, указанном на Рисунке 3.



Рисунок 3 - Часть скрипта со сменой уровня

Остальные сцены имеют другую логику и содержат несколько скриптов:

* Unit – общий класс для всех юнитов в игре;
* Character : Unit – класс контроля персонажа. Содержит методы управления персонаже, отображение анимации и другое;
* Boss : Unit – класс, отвечающий за функционирование босса;
* Warm – класс, отвечающий за функционирование ловушек;
* CameraController – класс, отвечающий за контроль камеры;
* PauseS – реализация паузы;
* ButtonManager – менеджер кнопок панелей;
* LoadScene – класс, отвечающий за логику загрузки сцен;
* LivesBar – отображение жизней в интерфейсе игры;

Все сцены находятся в папке Assets/Scenes.

Скрипты расположены в Assets/Scripts.

Также в Assets представлены вспомогательные папки с используемыми медиаматериалами: Fonts шрифты, Images изображения и анимации, Music музыка и Shablon с шаблонами объектов.

## Используемые функции

Некоторые функции требуют аргументов. Требования к аргументам описаны в таблице.

|  |  |  |
| --- | --- | --- |
| **Файл скрипта** | **Функция** | **Назначение** |
| CharacterController.cs  Основной скрипт работы всех уровней и персонажа. | Start() | Стандартная функция Unity. Выполняется 1 раз при первом запуске скрипта на сцене.  Задает основные объекты для других функций и некоторые первоначальные настройки, которые организованы сложнее обычной переменной или константы. |
| Update() | Стандартная функция Unity. Выполняется 1 раз на каждый кадр (фрейм) программы. Выполняет проверку всех необходимых функций как Flip(), Jump(), Trapped(), MoveRight() и т.д. |
| Flip() | Определяет направление движения персонажа для корректного отображения направления спрайта и анимации. |
| HitDelay() | Устанавливает задержку между соседними получениями урона от ловушки. |
| RemoveHitDelay() | Завершает задержку между соседними получениями урона от ловушки. |
| Trapped() | Проверка на столкновение с ловушкой. |
| Pause() | Приостановка игрового процесса и открытие игрового (на уровне) меню. |
| Jump() | Задает контроль прыжка – проверка на нахождение на земле и нажатие клавиши прыжка. |
| MoveRight() | Отвечает за перемещение персонажа вправо: до определенного значения расстояния от правого края окна перемещается сам персонаж, далее начинают перемещаться фон, ловушки и платформы, которые создают видимость движения персонажа.  Также реализует зацикливание фона и завершение уровня при касании финиша. |
| MoveLeft() | Отвечает за перемещение персонажа влево: до определенного значения расстояния от левого края окна перемещается сам персонаж, далее начинают перемещаться фон, ловушки и платформы, которые создают видимость движения персонажа. |
| GlobalNames.cs  Сохраняет данные о достижениях и громкости звуки на протяжении всех уровней. | Start() | Стандартная функция Unity. Выполняется 1 раз при первом запуске скрипта на сцене.  Запрещает уничтожение объекта музыки (с громкостью звука) при смене уровня. |
| Update() | Стандартная функция Unity. Выполняется 1 раз на каждый кадр (фрейм) программы.  При переходе в главное меню проверяет список полученных достижений и отмечает соответствующие в списке достижений в главном меню. |
| Mucis.cs  Задает громкость музыки и выбор песни для уровня. | Start() | Стандартная функция Unity. Выполняется 1 раз при первом запуске скрипта на сцене.  Задает громкость музыки, исходя из данных от GlobalNames.cs |
| MenuControls.cs  Отвечает за работу некоторых кнопок в главном меню. | PlayPressed() | Начинает новую игру при нажатии в главном меню кнопки «Новая игра». |
| ExitPressed() | Закрывает приложение. |

е

|  |  |  |
| --- | --- | --- |
| Unit  Содержит методы для дочерних классов | ReceiweDamage(Vector3 POZ) | Виртуальный метод, запускает метод Damage(). Требует вектор, содержащий позиции того объекта, который наносит урон. |
| isForce(Vector3 POZ) | Виртуальный метод(пустой) |
| Damage() | Метод уничтожающий объект. (переопределяемый) |
| Character  Управление персонажем | Start() | --//--//--//-- |
|  | Update() | --//--//--//-- |
|  | FixedUpdate() | Стандартная функция, вызывается с частотой фиксированных кадров (fixed framerate), если MonoBehaviour включен.  Используется FixedUpdate вместо Update когда необходимо работать с Rigidbody. |
|  | HitDelay() | Устанавливает задержку между соседними получениями урона от ловушки. |
|  | RemoveHitDelay() | Завершает задержку между соседними получениями урона от ловушки. |
|  | isForce(Vector3 poz) | Отталкивает персонажа в обратном направлении при касании ловушки/босса и получении урона. |
|  | ReceiweDamage(Vector3 poz) | Получение урона от ловушек/босса, отнимание жизней. |
|  | Dead() | Метод, отслеживающий смерть персонажа |
|  | Run() | Метод, отвечающий за движение персонажа по горизонтали. |
|  | Jump() | Реализация прыжка персонажа. |
|  | CheckGround() | Проверка – находится ли персонаж на платформе. |
| LoadScene  Отвечает за контроль загрузки уровней | LoadifDead() | При смерти персонажа, запускает уровень заново. |
|  | LoadifFinal() | Загружает следующий уровень, по завершению текущего. |
|  | Cat() | Загружает уровень босса с персонажем котом. |
|  | Dog() | Загружает уровень босса с персонажем псом. |
|  | Frog() | Загружает уровень босса с персонажем лягушонком. |
|  | LoadifMenu() | Загружает главное меню. |
| LivesBar  Отвечает за отображение жизней в интерфейсе игры. | Awaken() | Стандартный метод. Вызывается тогда, когда объект создаётся. |
|  | Start() | --//--//--//-- |
|  | Refresh() | Обновляет отображение жизней. |
| DieZone  Отвечает за смертельную зону, расположенную там, где нет платформы. Убивает персонажа. | OnTriggerEnter2D(Collider2D collider) | Одна из стандартных функций. Проверяет на соприкосновение коллайдеров объектов.  Проверяет на соприкосновение зоны с персонажем. |
| Warm  Отвечает за контроль ловушек | Awake() | --//--//--//--//--//-- |
|  | OnTriggerEnter2D(Collider2D collider) | Одна из стандартных функций. Проверяет на соприкосновение коллайдеров объектов.  Проверяет на соприкосновение ловушки с персонажем, если коснулся персонаж, нанести урон. |
| CameraController  Контроль камеры. | Awaken() | --//--//--//--//-- |
|  | Start() | --//--//--//--//-- |
|  | Update() | --//--//--//--//--  Проверяет позицию персонажа и перемещает камеру к этой позиции. |
| ButtonManager  Управляет кнопками различных панелей в игре. | PlayGame() | Продолжить игру(если стоит пауза) |
|  | MainMenu() | Загрузить главное меню во время паузы |
|  | OnTriggerEnter2D(Collider2D collider | --//--//--//--//--  Если касается персонаж финальной зоны, то активирует панель диалогов. |
| PauseS  Отвечает за паузу в игре | Update() | --//--//--//--//--  Отслеживает нажатие клавиши ESC, изменение переменной paused, активно ли DialPanel. В зависимости от ситуации, устанавливает паузу в игре или отключает её. |
|  | RevertPaused() | Изменяет значение переменной paused на противоположное. |
| Boss  Отвечает за контроль босса | Awake() | --//--//--//--  Запускает анимацию, получает необходимые компоненты. |
|  | Update () | --//--//--//--  Вызывает метод Move() |
|  | OnTriggerEnter2D(Collider2D collider | --//--//--//--  Проверяет: сверху или со стороны касается персонаж босса. Если сверху, получает урон босс, иначе персонаж. |
|  | Move() | Задаёт движение персонажа по определённой траектории. |
|  | ReceiweDamage(Vector3 poz) | Получение урона. |
|  | Dead() | Проверяет количество жизней босса, если равны 0, то загружает следующую сцену. |

## Сообщения программисту

Во время игры никаких выводимых сообщений не предусмотрено. Программа работает только внутри своего интерфейса и полностью стабильна.

При тестировании, отладке или разработки в Unity, используются некоторые сообщения DebugLog(). Также могут появляться сообщения самого Unity.