Министерство образования Республики Беларусь
Гомельский государственный технический
Университет им. П.О.Сухого

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

направление специальности 1-40 05 01-12 «Информационные системы и технологии в игровой индустрии»

Доклад по дипломной работе

Выполнил:
студент гр. ИТИ-41
Дубовцов И.Д.

[слайд 1] Приветствие

Я, студент группы ИТИ-41, Дубовцов Иван Дмитриевич, хочу представить вам дипломную работу на тему «Игровое 2*D* приложение в жанре аркадного симулятора жизни фермера в *top-down* проекции на платформе *Unity*».

[слайд 2] Цель разработки

Целью моей дипломной работы было разработать интерактивное игровое приложение в жанре аркадного симулятора жизни фермера в *top-down* проекции, в котором пользователи или игроки смогут ощутить фермерскую деятельность попробовав себя в роли фермера. Пользователям предстоит накапливать ресурсы, занимаясь посадкой, уходом и сбором урожая.

Созданное игровое приложения поможет с интересом провести время и развить долгосрочное планирование.

[слайд 3] Основные задачи в ходе разработки

Процесс создания такого игрового приложения был разбит на несколько этапов: анализ существующих методов реализации таких приложений, разработка архитектуры, реализация механик игрового приложения и создание графического интерфейса.

[слайд 4] Актуальность разработки

После аналитического разбора игровой индустрии можно сделать вывод, что разработка таких игровых приложений актуальна, так как рынок видео игр растет и видео игры являются одним из самых популярных видов развлечений в целом. Игры жанра аркадных фермерских симуляторов пользуются спросом среди широкой аудитории.

Также в приложении для общения с *NPC* используется технология *ChatGPT* от компании *OpenAI* для улучшения социального аспекта. Это способствует удержанию к игре интереса.

[слайд 5] Анализ представителей

В процессе анализа представителей было решено изучить два ярких продукта в схожем жанре. Ими являются: «*Stardew Valley*» и «*Farming Simulator*». В результате анализа можно выделить такие черты и механики, как сельскохозяйственный труд, содержание экономических элементов, ролевые игры.

[слайд 6] Концепт игры

После анализа представителей и аналитического обзора была придумана идея игры, которую можно описать следующим образом: игрок находится в параллельной вселенной в фермерской долине. Его целью является расширение своих имений для получения большего количества игровой валюты путем посадки и сбора урожая. При этом пользователь может общаться с игровыми персонажами, которые будут его направлять.

[слайд 7] Технологии разработки

Для реализации такого концепта были выбраны следующие инструменты разработки: *Unity, Visual Studio, Blender, Git*.

Visual Studio – это среда разработки, интегрированная в Unity, для удобства написания и отладки программного кода,

Blender – инструмент для создания 3*D* моделей, которые используются для 2*D* графики и создания карт нормалей и масок освещения.

Git – система контроля версий для отслеживания и сохранения изменений в процессе разработки.

В ходе анализа средств разработки было найдено два самых популярных игровых движка: *Unity* и *Unreal Engine*. Но для реализации концепта был выбран, именно, игровой движок *Unity,* который содержит обширный инструментарий для 2*D* игр. *Unreal Engine* больше подойдет для игр с 3*D* графикой.

[слайд 8] Функциональные возможности.

После первого этапа разработки приложения были выделены основные функциональные задачи, которые представляет приложение. Сюда входят следующие функциональные возможности: социальное взаимодействие, экономическая система, управление фермой, управление ресурсами, смена дня и ночи, сохранение и загрузка игрового состояния, настройка игровой среды.

[слайд 9] Схема взаимодействия игровых модулей приложения

Было принято решение разделить приложение на модули, которые выполняют определенный функционал приложения путем взаимодействия между собой. Таким образом, *TerrainModule* отвечает за функционал

управления фермой и взаимодействует с *StorageModule*, который отвечает за управление и использование ресурсов игрока. *TradeModule* отвечает за экономическую систему и взаимодействует с *StarageModule*, чтобы отображать стоимость покупки или продажи ресурсов игрока. *PlayerModule* отвечает за перемещение игрока. *DayNightModule* отвечает за смену дня и ночи, а *SoundModule* отвечает за проигрывание определенных звуковых эффектов.

[слайд 10] Схема взаимодействия модулей инициализации приложения

Модули инициализации игрового приложения нужны, чтобы инициализировать другие игровые модули. *InstallerModule* отвечает за поставку необходимых баз данных для работы модулей, а также параметры конфигурации приложения. *GameStateModule* отвечает за загрузку и сохранение игрового состояния, доставляет данные о сохраненном игровом состоянии в модули, которые в этом нуждаются.

[слайд 11] Схема сцен игрового приложения

Для удобства устройства игры было решено разделить игровой процесс на сцены. На начальной сцене игрок может настроить игровую среду, а также выбрать сохраненную игру или начать новую игру.

[слайд 12] Связь модулей с компонентами *Unity*

На этой схеме наглядно показано влияние функционала программных модулей приложения на игровые объекты на сцене. Путем написания игровых скриптов можно управлять компонентами игрового объекта и их свойствами.

[слайд 13] Схема работы модуля загрузки и сохранения игрового состояния

После запуска игры работает *DataLoader*, который загружает данные игрового состояния, после чего на начальной сцене игрок может воспользоваться сохраненными данными для запуска игры. В случае выхода из игры *DataSaver* сохраняет текущие данные игрового состояния локально на ПК пользователя. При загрузке игровой сцены данные об загруженном игровом состоянии помещаются в *DataContainer*, чтобы *GameStateGenerator* мог воспользоваться данными состояния для генерации загруженного уровня.

[слайд 14] Схема работы модуля социального взаимодействия

В игре игрок может открыть панель чата, чтобы начать общение с *NPC*. После отправки сообщения *ChatService* отправляет запрос к *OpenAI API*, после чего ожидает ответа. Данные об ответе отображаются в чат-панели, после чего *ChatOptimizer* начинает подсчет текущего размера контекста сообщений. В случае, если размер контекста больше заданного, игроку предстоит очистить историю сообщений, чтобы продолжить свое общение. Если размер контекста не превышен, то игрок может дальше свободно продолжить общение.

[слайд 15] Дальнейшее развитие

При дальнейшей поддержке программного продукта, имеется возможность добавления новых игровых механик и возможностей для улучшения пользовательского опыта. Планируется постоянная работа над оптимизацией работы приложения. Доступна дальнейшая интеграция с другими технологиями для расширения возможностей. И дальнейший выпуск готового приложения на различных платформах.

[слайд 16] Заключение

В процессе разработки была разработана концепция игры, проведено проектирование архитектуры приложения, были выбраны инструменты разработки *Unity, Visual Studio, Blender, Git* и язык программирования *C#* для программной реализации приложения.

В процессе выполнения дипломной работы было разработано игровое приложение «*Farmer's Valley*» под операционную систему *Windows*.

[слайд 17] Список опубликованных работ

Во время разработки дипломной работы, были опубликованы две работы:

– Искусственный интеллект в играх жанра *RPG,* «Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях»: научное издание, материалы XXVII Республиканской научной конференции студентов, аспирантов и молодых ученых, Гомель: ГГУ им. Франциско Скорины;

– Применение генеративного ИИ для игр в жанре *RPG*, «*E.R.A* – Современная наука: электроника, робототехника, автоматизация», материалы I Междунар. науч.-техн. конф, студентов, аспирантов и молодых ученых, Гомель: ГГТУ им. П. О. Сухого