УДК 004.514

ИГРОВОЕ ПРИЛОЖЕНИЕ СИМУЛЯТОР ФЕРМЫ «LITTLE ESTATE»

Е. И. ЮРЧИШКО
(Представлено: канд. техн. наук, доц. В. М. ЧЕРТКОВ)

Рассмотрены жанры игровых приложений. Определены задачи разработки компьютерной игры на Unity. Приведена упрощенная диаграмма классов игрового приложения «Little estate» жанра симулятор фермы. Представлены варианты взаимодействий с игроком.

Введение. Компьютерная игровая индустрия – это крупный рынок, постоянно развивающийся и разрастающийся, и является перспективным вектором развития, потому как каждый год появляются тысячи новых игр на различных платформах для распространения и продажи цифрового контента.

Игры, не только компьютерные, используются для развития различных навыков: воображения, творческого потенциала, реакции, логики, стратегического мышления. [1]. Под любой запрос существуют множество жанров и комбинируемых поджанров игр, сочетающих в себе различные механики для удержания внимания пользователя

Разработка игр на сегодняшний день в основном сопровождается использованием специального базового программного обеспечения, позволяющего использовать универсальные решения для создания игр, так называемые игровые движки. Такая основа позволяет использовать готовые структуры для создания сцен, использования игровой камеры, симуляции физики для игровых объектов. Игровые движки помогают оптимизировать процесс разработки, использовать стандартизированный и проверенный подход и позволить разрабатывать большие проекты даже одному разработчику.

На сегодняшний день игры жанра симулятор фермы снова начинают набирать популярность среди аудитории различных игровых площадок, например, Steam, VK Play, Good Old Games (GOG). Именно поэтому, переосмысление данного жанра и внесение новых механик позволит вовлечь заинтересованную аудиторию игроков. В современном производстве игр рамки каких-либо жанров всегда несколько размыты. Не всегда продукт полностью подходит под определенную категорию — очень часто он содержит в себе характерные черты нескольких из них [2].

Игровой проект «Little estate» – это игра жанра симулятор фермы, позволяющая игроку обустроить собственные землевладельческие угодья и развивать их на свое усмотрение, разбавляя игровой процесс посещением генерируемых комнат и побеждая враждебных существ для добычи новых ресурсов или валюты.

Для реализации задачи по проектированию игрового приложения «Little estate» использовался движок Unity, версии 2021.3.23f1. С его помощью удается создавать проекты под iOS, Linux и Windows, а также для разнообразных консолей (примеры – PlayStation, Xbox). Приложение имеет несколько тарифов – каждый предлагает отдельную функциональность. Для выпуска игр можно использовать предложение Personal, которое предназначается для частных лиц и небольших компаний.

Unity использует C# как основной язык для написания скриптов, что позволяет разрабатывать игры с высокой производительностью и гибкостью. C# тесно интегрирован в архитектуру Unity, что упрощает доступ к различным функциям движка. Также C# является языком объектно-ориентированного программирования, это особенно важно в разработке игр, где проекты могут становиться очень сложными.

Постановка задачи и проектирование игрового приложения.

Функциональное назначение разрабатываемой игры состоит в развитии у игрока стратегического мышления благодаря распределению и вложению собственных ресурсов в развитие фермы. Игра жанра симулятор фермы способствует тренировке творческого мышления и видения, потому как данный проект представляет собой инструмент для творчества.

К основным функциям игрового приложения относятся:

1 Распределение ресурсов экономическим путем для дальнейшего развития.

2 Механика добычи уникальных ресурсов в случайно генерируемых улицах города.

3 Система инвентаря, различные взаимодействия игрового персонажа под управлением пользователя с предметами окружения.

4 Диалоговая система.

5 Менеджер звуков.

Основой любой компьютерной игры является проекция игрока, выполняющая определенные команды. В проекте «Little estate» управление персонажем происходит от третьего лица. Для передвижения по поверхности спроектированных сцен был разработан скрипт контроллера движения персонажа. Пер-

сонаж игрока представляет из себя игровой объект с различными компонентами: Character Controller, Sprite Renderer и скриптами для реализации логики персонажа. Character Controller – это компонент в Unity, который предоставляет высокоуровневую систему управления персонажем, специально разработанную для обработки ввода игрока и физического движения персонажа в игровом мире. Он часто используется для создания персонажей в играх, особенно с видом от первого и третьего лица. В отличие от компонентов Rigidbody, Character Controller не использует физику Unity для перемещения. Вместо этого он управляется программно, что дает больше контроля над движением персонажа и позволяет избегать некоторых сложностей, связанных с физическим взаимодействием объектов.

Для анимации двумерного, но изометрического объекта используются спрайты, имитирующие отображение трехмерного объекта. Анимация для двумерных объектов в Unity может быть или на основе движения по так называемым «костям», то есть скелетной, или покадровой.

В проекте «Little estate» используется покадровая анимация для движения, атаки или других действий игрового персонажа. Чтобы управлять анимацией из кода, используется компонент Animator.

Во время передвижения по сцене игровой персонаж может находить предметы, входя в коллизию с которыми, он может начать взаимодействовать с ними, или помещать в свой инвентарь. Интерактивные предметы могут содержать в себе различную функциональность за счет реализации интерфейса IInteractable.

Рисунок 1. – Пример анимации персонажа

Разработка интерфейса в играх включает в себя пользовательское главное меню, а также интерфейс во время игрового процесса для отображения присутствующих механик, таких как характеристики персонажа, ресурсы игрока, инвентарь и тому подобное.

Инвентарь представляет из себя систему ячеек, в которых отображается некоторый предмет или их, так называемая, «пачка», то есть совокупность одинаковых игровых объектов. В одну ячейку инвентаря может поместиться ограниченное количество объектов, что должно отображаться цифрой, чтобы игрок мог контролировать степень занятости его «сумки».

Для перемещения предметов из инвентаря игрока в другой инвентарь игроку потребуется выбирать нужную ячейку, чтобы с ней производить действия. Выбранная ячейка должна выделяться некоторым образом, чтобы давать игроку представление о том, с какой именно ячейкой он в данный момент взаимодействует. В одну ячейку можно помещать только элементы одного типа.

Система диалога в игре также сопровождается определенным видом пользовательского интерфейса, при котором возникает затемняющее окно, ограничивающее взаимодействие игрока с миром. Оно не позволяет взаимодействие с другими элементами интерфейса, которые в данный момент должны быть недоступны. Кроме того, при проектировании такой системы стоит ввести ограничение на управление, при этом переназначив ответственность клавиш, чтобы использовать их для новых специфичных функций диалога.

Так как проект подразумевает наличие боевой системы, то для индикации прогресса требуется наличие определенных элементов интерфейса.

Направляющие подсказки – это элементы интерфейса, которые помогают пользователям быстрее распознавать те или иные доступные взаимодействия [3].

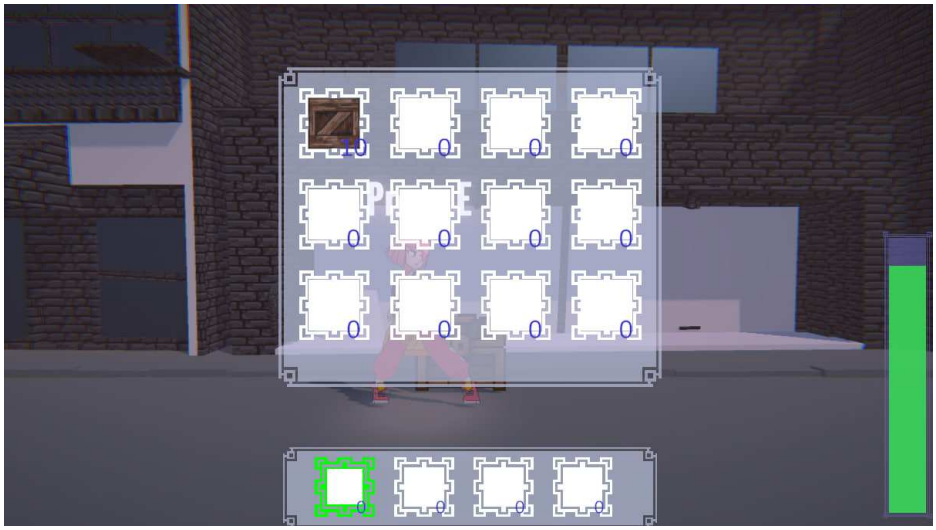

Рисунок 2. – Инвентарь игрового персонажа

Во время сражения у игрока отнимаются очки его основной характеристики. Для отслеживания оставшегося количества на экране должен располагаться индикатор прогресса.

Враги в играх тоже обычно имеют какую-либо прочность брони или характеристику здоровья, и при успешной атаке игрока теряют ее. Удары должны сопровождаться или анимацией попадания, или падением уровня шкал здоровья врагов, или «вылетающими» цифрами отображения урона. Также это относится к уничтожаемым ради выпадения ресурса предметам. Благодаря этому игрок может отследить успешность своих действий и, в случае чего, изменить местоположение персонажа, чтобы он начал успешное взаимодействие.

В проекте на сцене с генерируемыми комнатами появляются враждебные существа, являющиеся основополагающим компонентом боевой системы игры. Класс Enemy представляет из себя абстрактный наследуемый класс, в котором собрана общая логика для такого понятия как враг. Он содержит в себе реализацию системы здоровья врага, а также реализацию интерфейса IDamagable, отвечающего за наносимый урон объекту. Наследником класса врага и непосредственно существующим на сцене типом враждебного существа является летучая мышь. Скрипт BatEnemy описывает поведение летучей мыши-врага в игре. Летучая мышь патрулирует территорию, ищет игрока, преследует его и атакует. Скрипт управляет состояниями врага и его взаимодействием с окружающим миром, используя реализацию паттерна машина состояний.

Рисунок 3. – Боевая система

Реализация паттерна State Machine в классе BatEnemy управляет поведением летучей мыши в различных состояниях (патрулирование, преследование и атака). Этот подход упрощает управление слож-

ной логикой, разделяя поведение на четко определенные состояния. В классе BatEnemy определены следующие поля и состояния в типе перечислений: Patrol, Chase, Attack.

Звуковые файлы в играх используются в двух ситуациях: в качестве фонового саундтрека, или же как аудио эффект. Для удобного управления музыкой и звуковыми эффектами был написан класс-менеджер. Класс включает методы для воспроизведения звуков и музыки, а также для включения и отключения этих аудио компонентов в зависимости от настроек пользователя.

Для иллюстрации представления упрощенной внутренней структуры программы в виде компонентов и связей между ними была построена диаграмма классов.

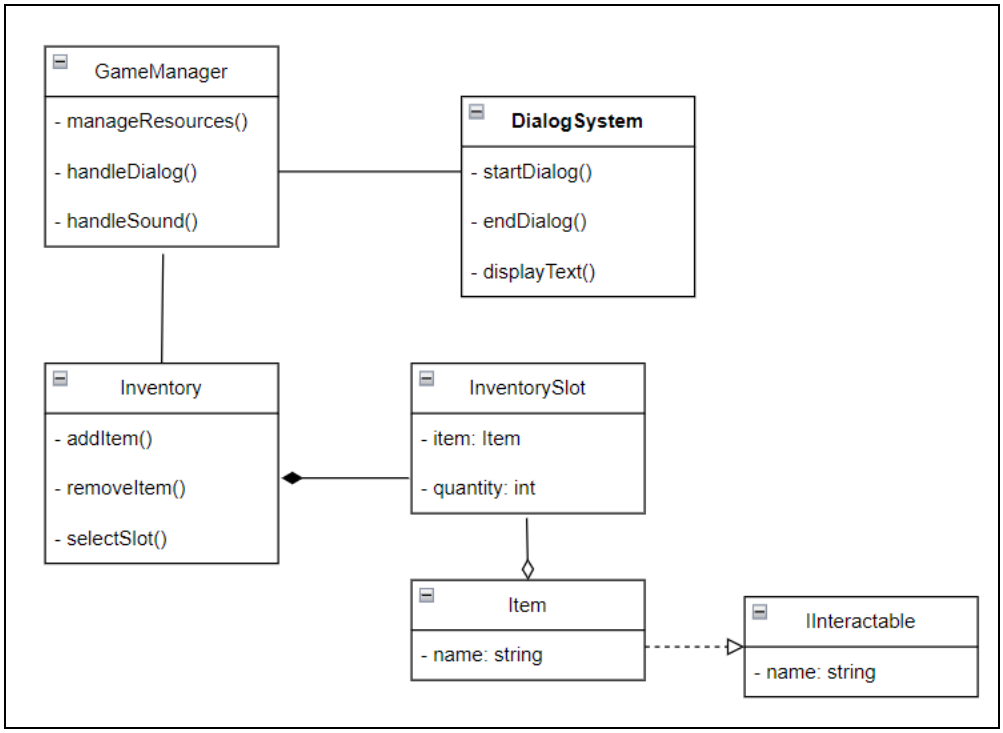

Рисунок 4. – Упрощенная диаграмма классов проекта

Класс GameManager управляет инвентарем и системой диалогов, потому между ними существует связь типа ассоциация. Inventory содержит множество InventorySlot, и их существование напрямую зависит от инвентаря. Это иллюстрируется стрелкой типа «композиция». InventorySlot содержит Item, и этот объект может существовать отдельно от слота. Такая связь называется агрегацией, и на диаграмме отображена стрелкой с незакрашенным ромбом на конце. Класс Item реализует интерфейс IInteractable для взаимодействия с игровым миром.

В Unity используются различные методы для сохранения информации, и один из таких – объекты типа ScripableObject. ScriptableObject – это класс, который позволяет хранить большое количество передаваемой информации независимо от образцов скрипта [4].

Объекты данного класса представляют собой специальные хранилища данных, в которых задаются контейнеры для их хранения, и из инспектора разработчик может их заполнить. Большой плюс такого использования ресурсов в том, что объект создается всего один раз, и после этого все классы и их экземпляры обращаются к нему при надобности. Этот подход позволяет выигрывать в количестве данных, которые будут занимать память. Многие классы проекта, включая звуковой менеджер, используют конфигурационные объекты.

Заключение. В ходе проектирования была разработана диаграмма классов, иллюстрирующая взаимодействия разработанных компонентов в игровом проекте. При разработке концепции графического интерфейса пользователя, а также при его проектировании на всех сценах игры, были соблюдены все требования. Анализ аналогичных программных продуктов на рынке позволил принять обоснованные решения относительно концепции и направления игры, избежать непопулярных и неэффективных решений в конкурирующих играх.

Программный продукт успешно прошел все этапы тестирования без выявленных дефектов. Игра «Little estate» показала себя, как готовый к эксплуатации программный продукт, стабильный в использовании и с интуитивно понятным интерфейсом.

ЛИТЕРАТУРА

1. Подвальный, М. А. Видеоигра [Электронный ресурс] // Большая российская энциклопедия. – Режим доступа: https://bigenc.ru/c/videoigra-915272 – Дата доступа: 14.05.2024.
2. Забубенин, В. Самые популярные жанры игр на ПК: очевидный топ-10 [Электронный ресурс] – Режим доступа: https://ichip.ru/gejming/igry/samye-populyarnye-zhanry-igr-na-pk-top-10-kategorij-853792 – Дата доступа: 14.05.2022.
3. 12 эффективных способов предотвращения ошибок в интерфейсах [Электронный ресурс] – Режим доступа: https://www.uprock.ru/articles/12-effektivnyh-sposobov-predotvrashcheniya-oshibok-v-interfeysah – Дата доступа: 14.05.2024;
4. Скриптуемый объект (ScriptableObject) [Электронный ресурс] – Режим доступа: https://docs.unity3d.com/ru/530/Manual/class-ScriptableObject.html. Дата обращения: 02.05.2024.