Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ОТЧЕТ

о преддипломной (производственной) практике

наименование вида и типа практики

на (в) ООО «Предприятие ВТИ-Сервис»

наименование предприятия, организации, учреждения

Студента 4курса, группы ПО-02б

курса, группы

Герасевой Ульяны Андреевны

фамилия, имя, отчество

Руководитель практики от предприятия, организации, учреждения

директор

должность, звание, степень

фамилия и. о.

Оценка _____

подпись, дата

Руководитель практики от университета

к.т.н. доцент

должность, звание, степень

Чаплыгин А. А.

фамилия и. о.

Оценка _____

подпись, дата

Члены комиссии

подпись, дата — фамилия и. о.

подпись, дата — фамилия и. о.

подпись, дата — фамилия и. о.

Курск 2024 г.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Игровой движок – базовое ПО компьютерной игры, которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений.

ИС – информационная система.

ИТ – информационные технологии.

ООП – объектно-ориентированное программирование.

ПО – программное обеспечение.

РП – рабочий проект.

ТЗ – техническое задание.

ТП – технический проект.

UML (Unified Modelling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

1 Анализ предметной области

1.1 История и описание Point-and-click игр

Point-and-click (point'n'click, point-n-click, с англ. — «укажи и щёлкни») это жанр видеоигр, где ключевым элементом игрового процесса является наведение курсора мыши на активные области и нажатие по ним. Чаще такие игры представлены в 2D, а перед игроком открывается область-локация, как правило, с видом сбоку. Пользователи могут отдавать приказы персонажу, собирать предметы, перемещать их и взаимодействовать с ними различным образом. Игры Point & Click любят разбавлять всяческими головоломками, интересным сюжетом, диалогами, отличным саундтреком и необычным графическим оформлением.

Главным образом в качестве манипулятора для управления данным действием используется компьютерная мышь, однако могут быть задействованы аналоги либо заменители мыши (джойстик, клавиатура). Типичным примером point-and-click является использование мыши в гипертекстовом документе, где нажатие по ссылке инициирует переход в другую область документа или в другой документ. Интерфейс point-and-click стал широко распространяться в компьютерных играх начиная с 1980-х годов с появлением мультимедийных домашних компьютеров (Amiga, Atari ST, IBM PC, Macintosh), которые уже могли поддерживать оконный интерфейс в своих операционных системах.

Одними из первых point-and-click стали использовать графические приключенческие игры как наиболее удобный способ взаимодействия пользователя с игровым миром. Этот метод произвёл небольшую революцию в данном жанре — произошёл основополагающий переход игры от первого лица к третьему, появился главный герой как обособленный игровой персонаж. Также наметился переход от командного режима управления персонажа (ввод глаголов-команд в текстовой строке) к управлению манипулятором. Самыми знаменитыми представителями жанра Point and Click можно назвать серии Syberia (Сибирь), The Curse of Monkey Island, Papers, Please, Botanicula и другие. Ниже вы можете найти самые лучшие игры с особенностью Point & Click на ПК (PC), PlayStation, XBox, айфон и андроид.

1.2 История и описание Приключенческих игр

Приключенческая игра (англ. adventure game) или квест (от англ. quest) — один из основных жанров компьютерных игр, представляющий собой интерактивную историю с главным героем, управляемым игроком. Важнейшими элементами игры в жанре квеста являются собственно повествование и исследование мира, а ключевую роль в игровом процессе играет решение головоломок и задач, требующих от игрока умственных усилий. Такие характерные для других жанров компьютерных игр элементы, как бои, экономическое планирование и задачи, требующие от игрока скорости реакции и быстрых ответных действий, в квестах сведены к минимуму или вовсе отсутствуют[1]. Игры, объединяющие в себе характерные признаки квестов и жанра action, выделяют в отдельный жанр — action-adventure.

Одним из видов приключенческих игр является графические квесты. Первые графические квесты появились ещё для 8-битных домашних компьютеров в начале 1980-х. Однако по-настоящему «графическими» они стали лишь в тот момент, когда произошёл отказ от текстового интерфейса и переход к так называемому «point-and-click» (то есть управлению с помощью указателя посредством стрелок клавиатуры, джойстика или мыши), появившемуся в 1985. Одними из популярных игр этого поджанра являются серии игр Monkey Island и Space Quest.

Развитие вычислительной техники и появление домашних компьютеров, отличающихся развитой графической системой (таких, как Apple II) послужило толчком к появлению индустрии компьютерных игр. Соответственно, жанр квестов также получил дальнейшее развитие. Квесты приобрели первые графические иллюстрации происходящего, которые поначалу были чисто декоративными. Зачастую графика была примитивной (например, векторная с заливкой) и это не требовало больших расходов памяти. Игрок по-прежнему управлял действиями персонажа, вводя последовательности команд на клавиатуре, а изображение было статичным и служило только для стимулирования воображения играющего.

Первые дискуссии о целесообразности применения графики в приключенческих играх относятся к 1983—1984 годам. Основным аргументом сторонников графики был: «Ничего в этом страшного нет, поскольку с одной стороны игры стано-

вятся привлекательнее, а расход памяти на графику компенсируется тем, что можно сэкономить на текстовом описании локаций». Противники этого подхода предупреждали, что усиление роли графики может привести к упрощению игр и к вырождению жанра, но технически графическое улучшение выглядело прогрессивно. Со временем сторонники графики победили и начали постепенно вытеснять текстовые игры.

Следующим этапом развития приключенческих игр стало изменение интерактивного взаимодействия игрока с виртуальным миром, когда вместо текстового ввода игрок мог кликнуть в некоторую область экрана мышью и получить результат — герой двигался в указанную точку, использовался указанный предмет и т. д. Так отпала необходимость в текстовом описании локаций — игрок мог «прощелкать» по всем объектам на экране и получить комментарий-описание что это такое.

К началу 1990-х годов адвентюры полностью изменились по сравнению с их оригинальными предшественниками начала 1980-х. Постепенное упрощение интерфейса привело к тому, что все многообразие глаголов (идти, взять, поговорить, положить и т. д.) свелось к одному слову «использовать»: «принять аспирин» → «использовать аспирин», «открыть дверь» → «использовать ключ на двери». Играть в такие игры стало проще, но была утрачена цель первых классических адвентюр — «установление контакта с программой и исследование её словаря». Следствием стало то, что упрощение стало влиять на логику игровых механик. Если у игрока оказывалось ограниченное число предметов и малое число команд, то он для прохождения перебирал все возможные варианты. Для того, чтобы сохранить простоту и сформировать новый игровой вызов, в игры стали добавлять элементы Action. Например, в игре Indiana Jones and the Fate of Atlantis в одном из эпизодов нужно похитить каменный диск прямо из-под носа у его владельца. Для этого нужно выключить свет, надеть на голову простыню, включить фонарик и изобразить из себя привидение, и пока оцепеневший от ужаса хозяин будет в шоке, нужно незаметно стащить диск. Как следствие, смещение в сторону экшен и дальнейшее развитие в этом направлении стало стимулировать появление игр жанра приключенческий боевик.

К первой половине 1990-х изменения коснулись в основном улучшения реалистичности, что делалось более широком применении видео и звуковых технологий.

Игры стали больше напоминать интерактивное кино, и для них стали привлекать профессиональных актёров.

1.3 Золотой век и упадок квестов

Соперничество двух крупнейших производителей квестов — Sierra On-Line и LucasArts благотворно сказывалось на самом жанре. В период с 1990 по 1998 год были выпущены лучшие игры из основных серий обеих компаний.

В очередной раз технический прогресс благотворно сказался на жанре компьютерных игр — с появлением звуковых карт появилась первая музыка, соответствующая атмосфере игры, а также озвучивание действий и событий. А с появлением таких ёмких носителей информации, как компакт-диск, стало возможным озвучивание диалогов персонажей.

Благодаря всему этому этот отрезок времени принято считать золотой эпохой графических квестов.

Однако появление первых графических акселераторов, а вслед за ними и первых трёхмерных игр послужило закатом эпохи квестов. Рынок требовал игры, демонстрирующей все возможности новых компьютеров.

Попытки сделать трёхмерные квесты имели лишь ограниченный успех, от такого внедрения технологии было больше вреда, чем пользы. В большинстве трёхмерных квестов был вид от третьего лица, и низкополигональные персонажи изображались поверх неподвижного двумерного фона. Неудачно расположенные (нередко неподвижные) точки обзора («камеры») дезориентировали игрока. Кроме того, с отказом от удобного и ставшего привычным режима point-and-click (с англ. — «укажи и щёлкни») играющему приходилось управлять персонажем с помощью стрелок клавиатуры, наблюдая зачастую, как тот бежит на месте, натолкнувшись на препятствие или невидимую стену. Существовали, правда, и полностью трёхмерные квесты с видом от первого лица.

Многие поклонники таких классических квестов, как Day of the Tentacle и Space Quest, не смогли принять эти трёхмерные новшества, а любители спецэффектов, как и прежде, обходили «занудные квесты» вниманием, предпочитая им всё более реалистичный 3D action и набирающий популярность жанр RPG. Испытывавшая

финансовые трудности Sierra On-Line продала своё подразделение по производству приключенческих игр, а LucasArts переключилась на более прибыльные игры, посвящённые сражениям во вселенной Звёздных войн.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки является задание на выпускную квалификационную работу бакалавра «Программная система для разработки приключенческих игр».

2.2 Цель и назначение разработки

Основной задачей выпускной квалификационной работы является разработка программной системы приключенческих игр для продвижения их популярности. Данный программный продукт предназначен для демонстрации практических навыков, полученных в течение обучения.

Целью данной разработки является создание программной системы для разработки приключенческих игр и популяризация этого игрового жанра.

Задачами данной разработки являются:

- проектирование интерфейса;
- разработка архитектуры приложения;
- проектирование игрового сценария;
- реализация взаимодействия приложения с пользователем;
- реализация графики приложения;
- реализация сохранений действий персонажа;

2.3 Требования пользователя к движку

Движок должен включать в себя:

- загрузку локаций;
- загрузку диалогов;
- загрузку объектов в их актуальном состоянии;
- сохранение состояний игровых объектов;
- возможность изменения сюжета игры;
- сохранение действий персонажа;
- возможность смены изображений игровых объектов.

Композиция шаблона игры, созданной на движке, представлена на рисунке 2.1.

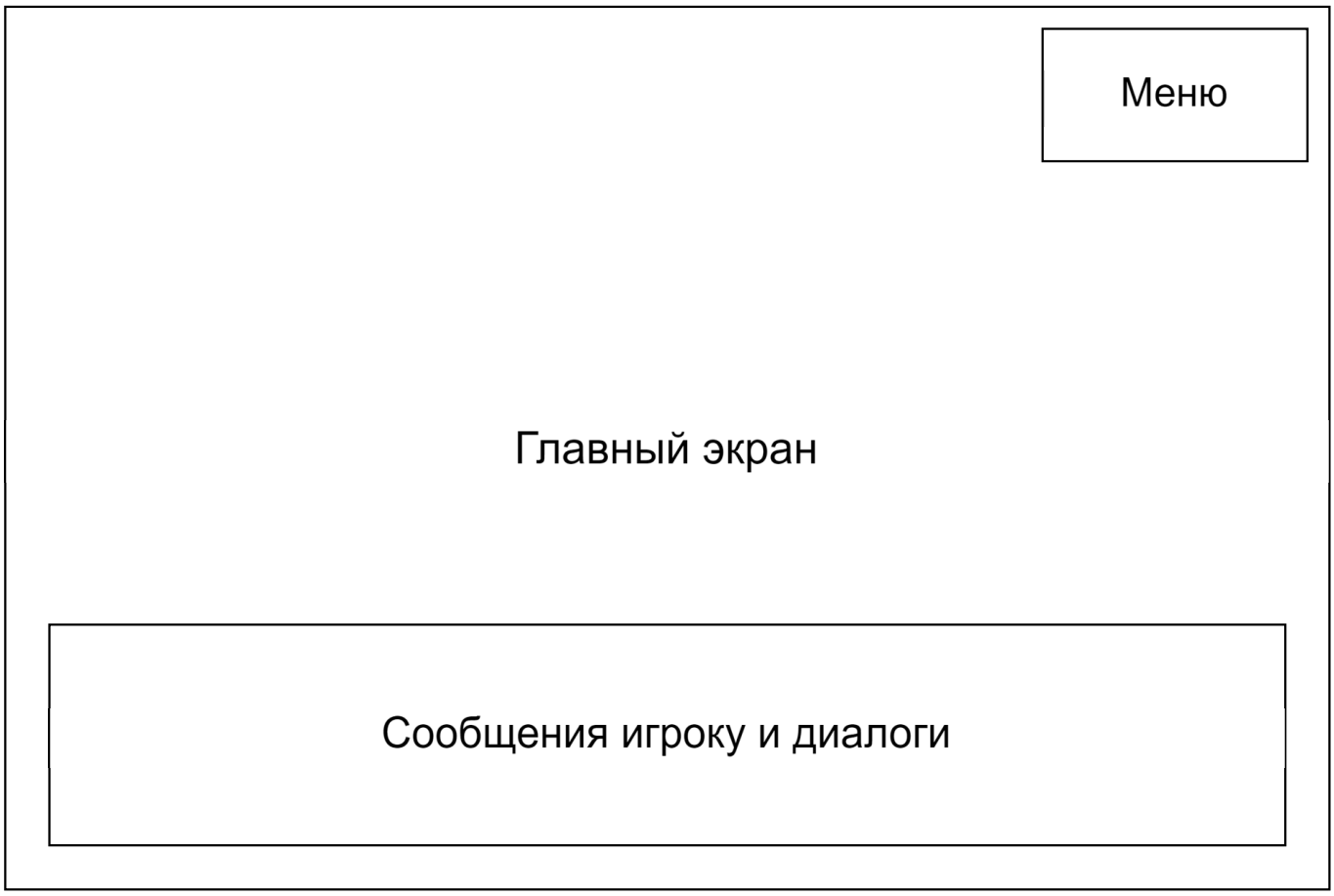

Рисунок 2.1 – Композиция шаблона интерфейса игры

2.4 Правила игры

Главный герой игры (ГГ) перемещается по локациям и выполняет квесты. Внутри каждой локации (кроме комнаты ГГ) имеется еще 2-3 подлокации. Начинается игра с предыстории в виде текста в диалоговой панели, который пользователь должен прочесть. Далее персонаж попадает в другие локации, перемещаясь по ним. Для того, чтобы пройти игру, нужно выполнить все квесты. Изначально не все локации доступны для посещения, а только те,что необходимы для текущего квеста. В игре присутствует цепочный механизм заданий, который предполагает последовательность действий (нельзя приступить к следующему заданию, не выполнив предыдущее). Процесс выполнения квеста заключается в поиске определенных предметов, взаимодействии с персонажами, предметами и локациями игрового ми-

ра. Выполнив необходимые действия ГГ дальше двигается по квесту. Игрок может взаимодействовать с игровым миром с помощью компьютерной мыши. При наведении на активный объект курсор мыши поменяет состояние и по нему можно будет понять, с каким объектом мы имеем дело. Доступны следующие объекты взаимодействия с игроком:

1. Дверь – при наведении курсора иконка указателя меняется на уменьшенное изображение двери. При нажатии на дверь меняется локация.

2. Сущность – при наведении курсор меняется. При нажатии открывается диалог.

3. Предмет – при наведении на него курсор меняется на руку. При нажатии предмет исчезает с карты и появляется в инвентаре.

4. Переход на другую локацию – при наведении иконка курсора меняется на стрелку в направлении доступной локации, в которую хочет перейти пользователь.

Инвентаря как такового нет. С ним нельзя взаимодействовать напрямую. Например, если при нажатии на какой-то объект требуется предмет в наличии у гг и его в инвентаре нет, то выйдет соответствующее сообщение, иначе – гг продвинется по квесту и получит другое сообщение. По завершению финального квеста игры и прохождению всех локаций пользователь видит развязку сюжета, это конец игры.

2.5 Сюжет игры

Главная героиня засыпает и оказывается в одной из локаций игры – в лесу. Там ее пугает обиатель, и она решается выйти из сна, но у нее не получается. Тогда она начинает изучать лес и сталкиваясь с его обитателями, пытается узнать у них, как ей проснуться. Персонажи дают ей квесты, например, принести что-то или чем-то помочь. По выполнении очередного квеста она получает полезную информацию, предмет или пропуск на следующую локацию. В конце концов, она оказывается у хозяина этих земель, который должен ей помочь проснуться.После краткого диалога он убивает героиню, и она просыпается в кровати.

2.6 Эпизод 1: Лес

Оказываемся в сумеречном лесу и осмариваемся. Неподалеку видим призрака, подходим к нему и нажимаем для взаимодействия. Девочка подмечает, призрак достаточно милый и не страшный. Начинается диалог с призраком, по окончанию которого призрак моментально увеличивается в размерах и строит страшную гримасу. Девочка пугается и убегает на другую локацию внутри леса - кладбище. Также, она понимает, что у нее не получается выйти из сна. На кладбище видим статую плачущего ангела. Когда подходим к ней, слышим звуки плача. Нажимаем на призрака и в ходе диалога выясняется, что статуя потеряла очень важную вещь в лесу и не может ее найти, тк обездвижена. Взамен на эту вещь статуя обещает помочь выбраться из сна. Не найдя эту вещь в данной части леса, движемся в единственную доступную - начальную, где мы встретили призрака. Увидев призрака, девочка опять испугалась, но не убежала. При осмотре этой локации понимаем, что нужного нам предмета здесь нет, однако достаточно большую часть карты занимает огромный призрак, которого нужно как-то прогнать или уменьшить. Девочка не придумала ничего лучше, как напугать его. Кликаем по ней и смотрим, как она увеличивается в размерах и меняется в лице. Тем временем, по мере увеличения девочки, призрак уменьшается и в конце концов приходит в прежнюю милую форму, а вскоре и совсем убегает. Замечаем, что он закрывал собой тот самый предмет, который мы искали. Забираем его и приносим статуе. Та нас благодарит и говорит, что единственный, кто может помочь ей выбраться из сна - правитель этих земель, а дорогу к нему знает только лис, который спит в этом лесу. Нам открывается путь на следующую локаци этого леса - логово лиса. Зайдя на нее видим большую морду спящего животного. Для того чтобы его разбудить, нужно 3 раза нажать на него. Лис соглашается помочь девочке и провести ее к правителю, но взамен он просит помочь ему с его проблемой: в него что-то залезло и постоянно мешает жить. Лис предлагает девочке залесть к нему в пасть и посмотреть, что там. Нажимаем на пасть лиса и оказываемся в новой локации. Основные объекты и способы взаимодействия с ними (при нажатии):

1. Призрак - говорить.
2. Статуя - говорить (получить квест), отдать предмет.

3. Венок – взять, отдать статуе.

4. Главный герой - вырасти и напугать призрака.

5. Переход на другую локацию – перейти к призраку, статуе, лису или в пасть к лису.

6. Лис - разбудить, говорить.

2.7 Интерфейс пользователя

На основании анализа предметной области в программе должны быть реализованы следующие прецеденты:

1. Перемещение по локациям;
2. Перемещение персонажа по игровой области;
3. Просмотр кат-сцен;
4. Просмотр диалогов;
5. Взаимодействие с игровыми объектами при помощи компьютерной мыши.

Таким образом, на рисунке 2.2сформированы следующие действия пользователя и их последствия.

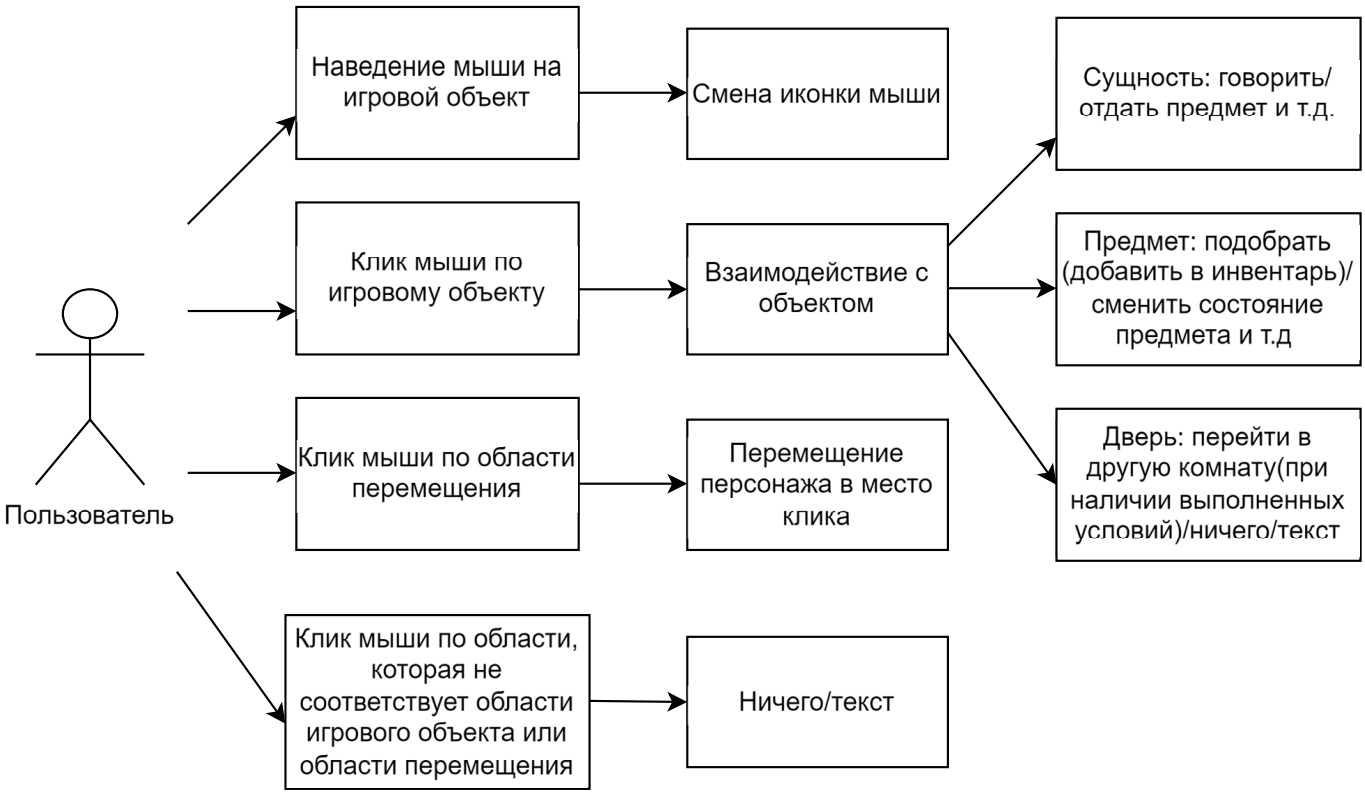

Рисунок 2.2 – Шаблон интерфейса игры

2.8 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Необходимо спроектировать и разработать приложение, которое должно способствовать продвижению компании на рынке.

Игровой движок – базовое ПО компьютерной игры, которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений. В дополнение к многократно используемым программным компонентам, игровые движки предоставляют набор визуальных инструментов для разработки. Эти инструменты обычно составляют интегрированную среду разработки для упрощённой, быстрой разработки игр на манер поточного производства.

3.2 Обоснование выбора технологии проектирования

На сегодняшний день информационный рынок, поставляющий программные решения в выбранной сфере, предлагает множество продуктов, позволяющих достигнуть поставленной цели – разработки игрового приложения.

3.2.1 Описание используемых технологий и языков программирования

В процессе разработки игрового движка используются программные средства и языки программирования. Каждое программное средство и каждый язык программирования применяется для круга задач, при решении которых они необходимы.

3.2.2 Язык программирования C#

C# - объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов

(в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей. C# является объектно-ориентированным и в этом плане много перенял у Java и C++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей.

3.2.2.1 Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, .NET MAUI). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

1. Поддержка нескольких языков.
2. Кроссплатформенность.
3. Мощная библиотека классов.
4. Разнообразие технологий.
5. Производительность.

3.2.2.2 Преимущества использования и изучения C#

C# является управляемым языком программирования, что позволяет разработчику не следить за выделением и использованием памяти. Для этого существует CLR (Common Language Runtime) – виртуальная машина, которая занимается запуском

приложения, а также управлением памятью. Также C# является строго типизированным и объектно ориентированным языком, что позволяет использовать ООП в его классическом виде. Здесь нет множественного наследования классов, что упрощает понимание ООП, но есть множественная реализация интерфейсов, что дает большую гибкость для разработчиков. Большое сообщество и универсальность языка дают большое поле для деятельности. Как уже было указано ранее, вы можете разрабатывать веб-приложения, сложные микросервисные платформы, игры, а так же мобильные приложения. Здесь действительно серьезный инструментарий для разработки, такие IDE как Visual Studio или JetBrains Rider. Наличие огромнейшего разнообразия библиотек на все случаи жизни, от обратки изображений и видео, до нейросетей. А кроссплатформенность дает возможность писать код как на Windows, так и на macOS и Linux. Как уже было сказано ранее, C# является языком программирования общего назначения, а значит покрывает большое количество задач и областей, а именно:

1. Web – разработка web-приложений и сервисов для платформ macOS, Windows, Linux и Docker.
2. Mobile – разработка единой кодовой базы для построения нативных приложения для iOS и Android.
3. Desktop – разработка нативных приложения под Windows и macOS.
4. Microservices – разработка независимых компонентов запускаемы в Docker контейнерах.
5. Cloud – использование существующих облачных решений или создание собственных. C# поддерживается большинством облачных платформ, такими как Azure и AWS.
6. Machine learning – разработка приложений искусственного интеллекта и машинного обучения, решающие проблемы машинного зрения, обработки речи, моделей предсказания, и тд.
7. Game development – разработка 2D и 3D игра для самых популярных десктопных и мобильных платформ.
8. Internet of Things (IoT) – разработка приложений для интернета вещей, имеющие поддержку Rasbery Pi и других одноплатных компьютеров.

Исходя из вышеперечисленных областей применения видно, что платформа .NET и язык программирования C# покрывают большой спектр проектов на рынке. Это говорит нам о том, что изучив язык программирования C# с легкостью можно найти проект на любой вкус.

3.2.2.3 Описание движка

AdventureGame - главный класс движка. Создается класс-наследник MyGame. Здесь будут храниться глобальные переменные конкретной игры (например флаги каких-то событий или квестов). Все функции движка там: добавить комнату, перейти в комнату, поместить персонажа/предмет, установить главного персонажа, удалить персонажа, предмет, запустить/остановить сценарий, передать управление в другой сценарий, разговор персонажа, взять предмет. То есть это все то, что должно быть доступно в любом сценарии. Поэтому, надо делать класс AdventureGame статическим.Саму игру мы задаем в конструкторе это класса как множество комнат.

Комната (Room) представляет собой совокупность объектов, а также хранит свою графику. Каждый объект может иметь свой небольшой управляющий сценарий. Эти сценарии работают параллельно, например, несколько объектов могут двигаться одновременно, или происходит одновременная анимация (качается дерево). В этих же сценариях можно управлять звуком (в будущем). Каждый сценарий запускается один раз и далее повторяется в цикле.Комната имеет отдельные сценарии для входа и выхода (то есть запускается один раз при входе героя в комнату и при выходе).К комнатам идет обращение по имени (задается при добавлении комнаты в движок). Сценарий входа в комнату может проверять какие-то игровые ситуации и настраивать комнату перед отображением. Сценарий выхода может останавливать какие-то вещи. Каждый объект в комнате имеет имя и сценарий по взаимодействию с ним по клику мыши (например открыть дверь).Также есть еще сценарий, когда герой пересекает объект (подходит к объекту). Здесь происходит перемещение персонажа в другую комнату (указывается в какую комнату). В другой комнате аналогично настроен сценарий на обратный проход (там другая дверь, другой объект).

Персонажи (Actor) имеют свою анимацию движения по комнате. По клику мыши в точку X,Y (или по программно заданным координатам) происходит автомати-

ческое движение. В комнате также задаются области (как множество прямоугольников), где происходит движение персонажей. Например, если в комнате в центре препятствие, то нужно 4 прямоугольника. Каждый прямоугольник должен касаться друг друга и не должен пересекаться с другими. Каждый персонаж Actor может иметь набор предметов или инвентарь.

Текст добавляется методом SayLine (например в сценарии взаимодействия). Объекты расставляются в комнате по координатам. При взаимодействии с объектом персонаж находится в определенном месте (эта позиция задается для объекта) - например, с левой стороны двери. Состояние персонажа (куда он смотрит после того как подошел в заданную позицию) задается в сценарии взаимодействия (запускается после того как персонаж пришел в заданную позицию). Объект может иметь несколько состояний (одно минимум). Каждое состояние сопоставлено с о спрайтом или анимацией. Например, закрытая и открытая дверь. Взять объект - PickUpObject - объект попадает в хранилище персонажа.

Анимацию задаем как последовательность кадров с заданной скоростью. Автоматическое движение объекта задаем в сценарии.

Сценарий запускается (StartScript), и затем работает до тех пор пока не будет остановлен (StopScript). В сценарии задаем шаги движения, между которыми вызываем Yield (передать управление другому потоку). То есть сценарии - это параллельные потоки. Глобальные переменные могут быть в классе конкретной игры или в статическом классе Globals. Чтобы к ним обратиться, функция сценария должна также быть в классе игры. Локальные переменные могут быть внутри конкретной комнаты, тогда локальные сценарии находятся в классе комнаты.

Сценарий - это любой метод (например у MyGame или у конкретной комнаты (класса, потомка от Room). То есть мы в движок передаем метод (через делегат), а уже сам движок делает все что нужно (запускает поток). У наследника указываются только свои поля. Сценарии размещаются в зависимости от того, что там делается. Если просто перемещение, смена состояния, смена комнаты - тогда в классе конкретной комнаты (потомка Room).

Отрисовка всех объектов происходит с помощью таймера и события Form_Paint. Таймер обращается к GraphicsManager, который меняет состояние

объектов когда это нужно. Это происходит каждый тик таймера. Затем событие Form_Paint с помощью Graphics отображает измененные объекты на форме. На выходе мы получаем актуальное состояние каждого объекта, которое можем наблюдать в реальном времени. Это может быть анимация перемещения, появления или исчезновения объекта.

3.3 Диаграмма компонентов классов

Диаграмма компонентов описывает особенности физического представления разрабатываемой системы. Она позволяет определить архитектуру системы, установив зависимости между программными компонентами, в роли которых может выступать как исходный, так и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы, а также зависимости между ними. На рисунке 3.1 изображена диаграмма компонентов для проектируемой системы. Она включает в себя основной класс движка игры AdventureGame и производные от него классы, класс Object с наследниками и их параметрами (полями и методами).

3.4 Описание классов

GraphicsManager - класс, управляющий спрайтами. Содержит в себе:

- List<Sprite> sprites - список спрайтов;
- double scalling_coef - коэффициент приближения;
- AddSprite(Sprite sprite, int x, int y) - добавляет в список спрайт, сохраняет координаты;
- ChangeSprite(Sprite sprite1, Sprite sprite2) - меняет местами спрайты в списке;
- DelSprite(Sprite sprite) - удаляет спрайт из списка;
- DelAll() - очищает список спрайтов;
- UpdateGraphics(Graphics g) - с помощью Graphics добавляет все спрайты из списка на форму.

Sprite - класс, хранящий в себе изображения игровых объектов Image img.

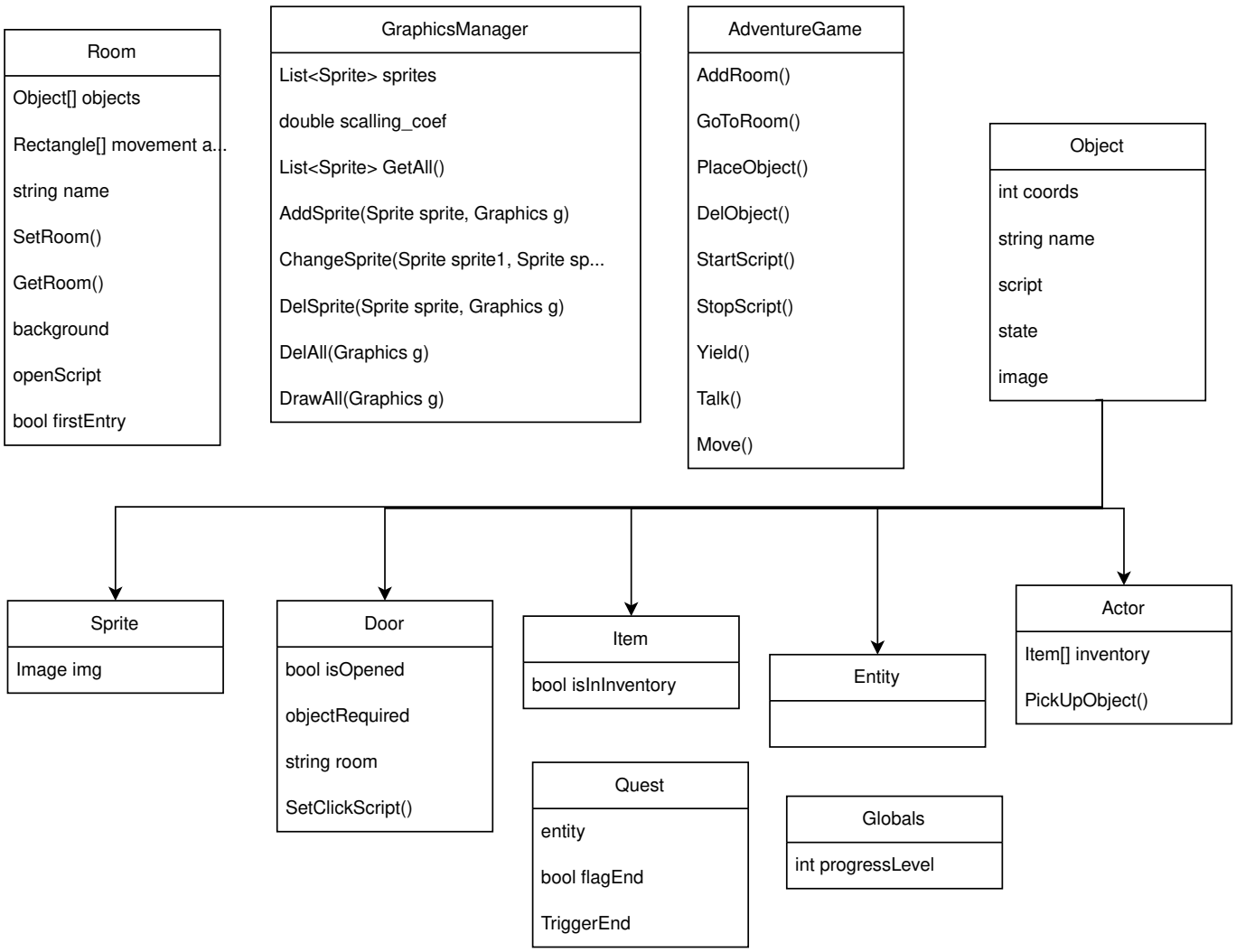

Рисунок 3.1 – Диаграмма компонентов

Object - абстрактный класс, от которого наследуются классы Sprite, Item, Entity, Actor, Door.

Entity - класс сущностей. Потомок класса Object. Сущность - игровой объект, который считается одушевленным, то есть с ним можно поговорить и нельзя подобрать в инвентарь, в отличие от объекта Item.

Item - класс предметов. Предмет - это игровой объект, который является неодушевленным, то есть вещью, которую можно взять в инвентарь. С вещью нельзя разговаривать, в отличие от объекта класса Entity, но она также является потомком класса Object.

Actor - главный персонаж игры, которым управляет пользователь. Именно он находится в центре сюжета. Является наследником Object, так как имеет схожие свойства.

Door - еще один наследник класса Object. Является по своей сути объектом, реализующим переход между комнатами(Room).

- bool isOpened - состояние двери (открыта/закрыта);
- objectRequired - условия, необходимые для того, чтобы дверь открылась. Проверяются при нажатии на дверь;
- string name - название комнаты, в которую ведет дверь;
- SetClickScript() - события, которые произойдут, когда польователь кликнет на дверь;

Room - комната(подлокация), в которой происходит сюжетное действие и игровой процесс. Содержит следующие поля и методы:

- Object[] objects - список всех объектов, которые размещены в данной комнате;
- Rectangle[] moveArea - область, по которой может перемещаться персонаж (Actor). Определяется набором прямоугольников, которые не должны пересекаться;
- string name - название комнаты;
- bool firstEntry - показывает, находится ли персонаж в данной комнате впервые;
- SetRoom(string roomName) - устанавливает комнату со всеми ее объектами и их состояниями;
- GetRoom(string roomName) - получает комнату со всеми ее объектами и их состояниями;

Globals - в данном классе находятся глобальные переменные (флаги) текущей игры. Например, int progressLevel в виде простого числа показывает прогресс игрока в игре.

AdventureGame - абстрактный класс, управляющий игрой. Имеет следующие методы:

- AddRoom(string roomName) - добавить комнату;
- GoToRoom(string roomNAme) - перейти в комнату;
- PlaceObject(Object object, int x, int y) - поместить объект на карту по координатам x и y;
- DelObject(Object object) - удалить объект;

- StartScript(???) - запускает скрипт;
- StopScript(???) - останавливает скрипт;
- Yeild(???) - передает управление другому скрипту;
- Talk(???) - говорить с персонажем. Отображает текст в текстовом поле внизу экрана.
- Move() - перемещение объектов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фримен, А. Практикум по программированию на JavaScript / А. Фримен. – Москва : Вильямс, 2013. – 960 с. – ISBN 978-5-8459-1799-7. – Текст : непосредственный.

2. Бретт, М. PHP и MySQL. Исчерпывающее руководство / М. Бретт. – Санкт-Петербург : Питер, 2016. – 544 с. – ISBN 978-5-496-01049-8. – Текст : непосредственный.

3. Веру, Л. Секреты CSS. Идеальные решения ежедневных задач / Л. Веру. – Санкт-Петербург : Питер, 2016. – 336 с. – ISBN 978-5-496-02082-4. – Текст : непосредственный.

4. Гизберт, Д. PHP и MySQL / Д. Гизберт. – Москва : НТ Пресс, 2013. – 320 с. – ISBN 978-5-477-01174-2. – Текст : непосредственный.

5. Голдстайн, А. HTML5 и CSS3 для всех / А. Голдстайн, Л. Лазарис, Э. Уэйл. – Москва : Вильямс, 2012. – 368 с. – ISBN 978-5-699-57580-0. – Текст : непосредственный.

6. Дэкетт, Д. HTML и CSS. Разработка и создание веб-сайтов / Д. Дэкетт. – Москва : Эксмо, 2014. – 480 с. – ISBN 978-5-699-64193-2. – Текст : непосредственный.

7. Макфарланд, Д. Большая книга CSS / Д. Макфарланд. – Санкт-Петербург : Питер, 2012. – 560 с. – ISBN 978-5-496-02080-0. – Текст : непосредственный.

8. Лоусон, Б. Изучаем HTML5. Библиотека специалиста / Б. Лоусон, Р. Шарп. – Санкт-Петербург : Питер, 2013 – 286 с. – ISBN 978-5-459-01156-2. – Текст : непосредственный.

9. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

10. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

11. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

12. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

13. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.