СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данном курсовом проекте предполагается реализовать игру с графическим интерфейсом при помощи инструментальной среды программного обеспечения, а именно, языка программирования C++. Ранее упомянутый язык разработан Бьерном Страуструпом в 1979 году и остается одним из мощнейших и популярных языков программирования.

Язык программирования C++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. C++ унаследовал от Си богатые возможности по работе с памятью, поэтому нередко C++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит и антивирусов. К слову сказать, ОС Windows большей частью написана на C++. Но только системным программированием применение данного языка не ограничивается. C++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где C++ тоже нашел свое применение. И даже в веб-разработке также можно использовать C++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения.Его популярность была вызвана объектно-ориентированностью языка. В общем C++ – язык широкого пользования, на котором можно реализовать практически любые виды программ [1].

Существует множество графических библиотек для написания игр, например, SFML, SDL, OpenGL, DirectX. В данном курсовом проекте используется одна из самых популярных графических библиотек – Simple and Fast Multimedia Library – простая и быстрая мультимедийная библиотека. SFML предоставляет набор классов для отображения простых фигур. Каждый из типов фигур – это отдельный класс, но все они происходят из одного базового класса и потому имеют набор одинаковых методов. Каждый класс добавляет специфичные для него свойства: радиус для круга, размеры для прямоугольника, вершины многоугольника и другое. Использование средств библиотеки SFML позволяет уделить большее внимание игровым алгоритмам, поскольку не требуется отдельно создавать функции для вывода графики и звука, и регистрации событий, обеспечивает простой интерфейс для разработки.

SFML является кроссплатформенной библиотекой, а это значит, что приложения будут работать на большинстве операционных систем (Windows, Linux, Mac OS X). SFML поддерживает большое число языков программирования (официально Си и .Net подобные языки, а благодаря различным сообществам ещё и такие как Java, Ruby, Python, Go, и другие) [2].

На данный момент самой популярной отраслью программирования являются компьютерные игры. Развиваясь с каждым годом все быстрее, становясь, при этом, более реалистичными и полифункциональными, и, как следствие, вовлекающие еще большее количество людей.

Основное предназначение реализуемого программного продукта – позволить работать воображению человека, ассоциировать себя с главным героем и внимательно следить за сюжетом, быть творцом в игровом мире, и с удовольствием и интересом провести свободное время.

Данный программный продукт рассчитан на пользователей подросткового возраста. Компьютерные игры обладают мощными развивающими характеристиками. Они влияют на развитие всех познавательных процессов: мышления, внимания, памяти и, конечно же, реакции. В современном мире самой крупной игровой платформой стали персональные компьютеры, которые, благодаря последним технологиям, достаточно производительны и многофункциональны. К тому же имеют свободный доступ в Интернет, который, в свою очередь, открывает двери в мир многопользовательских игр, позволяющих играть совместно с другими игроками.

Одним из параметров, по которому пользователь может выбрать себе компьютерную игру является жанр. Он определяется геймдизайнером, который может вознамериться устрашить игрока, сделать вызов его интеллекту, поразить красотой игровых сцен и др. Основными жанрами игр являются: приключенческая игра – главной частью игры является история, экшен – игра, характеризующая частым и активным нажатием кнопок управления, стратегическая игра – необходимость игроку делать нетривиальный выбор, компьютерный симулятор – игрок делает множество упражнений и оттачивает свою технику, головоломка – требует аналитического мышления, обучающая игра – игрок обучается во время выполнения каких-либо действий в игре. Данный список не является полным, и, в тоже время, игры могут комбинировать несколько жанров, что делает игровую индустрию еще интереснее и шире, а благодаря современным технологиям, игры, даже одинаковых жанров, поражают многообразием и предлагают пользователю уникальные игровые миры и персонажей.

1.ОБЗОР ИСТОЧНИКОВ

Ниже представлены основные термины в программировании, так как без понимания их невозможно будет выучить ни один из языков программирования:

Программирование – процесс и искусство создания компьютерных программ с помощью языков программирования. В узком смысле слова, программирование рассматривается как кодирование – реализация одного или нескольких взаимосвязанных алгоритмов на некотором языке программирования. В более широком смысле, программирование – процесс создания программ, то есть разработка программного обеспечения. Большая часть работы программиста связана с написанием исходного кода на одном из языков программирования.

Программа – данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма. **Программное обеспечение**– совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ. **Разработка рограммного обеспечения** – это проектирование, написание, тестирование и поддержка компьютерных программ с целью решения задач для множества пользователей; это создание надежных защищенных решений, которые выдержат испытание временем и справятся с некоторыми не известными заранее задачами, лежащими в области, близкой к очевидным исходным задачам. **Программный модуль** – программа или функционально завершенный фрагмент программы, предназначенный для хранения, трансляции, объединения с другими программными модулями для загрузки в оперативную память. **Объектно-ориентированное программирование** – метод построения программ как совокупностей объектов и классов объектов, которые могут вызывать друг друга. **Компиляция** – трансляция программы с языка высокого уровня в форму, близкую к программе на машинном языке.

ОЗУ (Оперативное запоминающее устройство – энергозависимая память в которой хранятся данные и команды необходимые процессору для выполнения им операций. **CPU** (центральное обрабатывающее устройство) – исполнитель машинных инструкций (кода программ). **ALU** – блок процессора, который служит для выполнения арифметических и логических преобразований над данными. **Разрядность процессора**— способность одновременно обрабатывать какое-то количество бит.

Переменная (Variable) – это область памяти, которая хранит в себе некоторое значение, которое можно изменить. Все последующие присвоения новых значений этой переменной, не считаются инициализацией. **Идентификатор** — последовательность символов, которые используется для именования членов, таких как переменные, методы, параметры, а также множество других программных конструкций [3].

1.1.Обзор аналогов

В настоящее время достаточно сложно создать интересную захватывающую игру не похожую на проекты прошлых лет. В первые годы развития игровой индустрии разработчики создавали игры, которые представляли собой один конкретный жанр. Позже, когда появилось достаточно однотипных игр, постепенно начали смешивать жанры с целью разнообразить свои проекты. Также благодаря развитию компьютерного моделирования, позволяющего создавать всевозможные иллюстрации, игры стали красочнее и интереснее. В итоге, с помощью вышеуказанных, а также других способов, разработчики получили огромные возможности при разработке своих игр. В данной курсовой работе проектируется 2D игра, которая, в свою очередь, является уникальной в своем жанре благодаря авторской логике персонажей, прорисовке изображений и игровой механике.

Несмотря на это у этой игры есть свои аналоги. В основе данного курсового проекта находится популярная трилогия «Dark Souls». С виду «Dark Souls» кажется вполне обычной. Это темное фэнтези с драконами и скелетами, абстрактным сюжетом про заблудшую душу и многовековое пророчество. Пробираясь сквозь поросшие мхом руины пользователю необходимо уничтожить все, что попадается на пути. В этом незатейливом процессе и кроется главная особенность игры. Постоянно вступать в рукопашную схватку с применением колюще-режущего оружия. В роли противников выступают огромные чумные крысы, ходячие мертвецы, гиганты с дубинами, сороконожки с клыками и драконы всех видов. Каждое противостояние выстроено с математической тщательностью, и чтобы победить, необходимо знать особенности каждого монстра и уметь молниеносно реагировать на его приемы. Это подобно шахматному бою, который длится десятки часов подряд. Это связано с тем, что погибнуть игрок может не только от меча, но и потому, что сорвался с обрыва, утонул, сгорел, был задавлен валуном или распилен огромным раскачивающимся топором. Мир «Dark Souls» создан лишь для того, чтобы испытать игрока на выносливость и реакцию.

Данный курсовой проект был вдохновлен этой игрой, ее атмосферой, механикой, окружением и эмоциями, после очередной трудной победы над боссом. Изначально «The Quest for the treasure unseen» проектировалась как достойная адаптация «Dark Souls» в 2D игру с элементами платформера. Благодаря обширному опыту в игровой индустрии, разработчику данного курсового проекта удалось передать боевую механику и главную особенность Темных Душ – трудность прохождения, а также дополнить новыми возможностями, что делает ее не очередным аналогом популярной игры, а достойным конкурентом другим играм-платформерам. Кроме того, «The Quest for the treasure Unseen» имеет более приятный интерфейс и окружение, а также сбалансированное увеличение сложности. Данные особенности позволяют продвинуть проект в список популярных компьютерных игр и привлечь максимальное количество пользователей.

1.2.Основные технологии

В процессе разработки программных систем используются различные технологии программирования. **Технологии программирования** – это способы создания программ. Эти способы включают в себя как определённые знания (например, знание языка программирования), так и определённые инструменты (например, средства разработки программ), другими словами это совокупность знаний и способов, использование которых приведёт к созданию нужной программы – от идеи до результата.

Развитие технологий программирования – это эволюция способов разработки программ. Эту эволюцию можно разбить на следующие этапы: стихийное программирование, структурное программирование, модульное программирование, объектно-ориентированное программирование, компонентный подход и CASE-технологии.

В данном курсовом проекте за главную технологию взято объектно-ориентированное программирование (ООП). Суть ООП заключается в представлении программы в виде совокупности объектов. Каждый из объектов имеет свои свойства (характеристики) и методы (функции). При этом программисту часто не обязательно знать, как устроен объект. Достаточно только общего описания свойств и методов. роме того, объектный подход предлагает новые способы организации программ, основанные на механизмах наследования, полиморфизма, композиции. Это позволяет существенно увеличить показатель повторного использования кодов и создавать библиотеки классов для различных применений. Один из объектно-ориентированных языков программирования – это C++, на котором, собственно, и разработан курсовой проект [4].

Дополнительной библиотекой C++ является SFML. Библиотека SFML позволяет разработать игровой проект, организовать вывод графических изображений и звуков, при этом уделить значительное внимание алгоритмам игрового процесса. Значимым преимуществом графической библиотеки является её кроссплатформенность, которая позволяет написать проект в одной среде и использовать в других, перекомпилировав программу, не изменяя код. В состав SFML входит пять модулей: system, window, graphics, audio и network. Основной функцией SFML является функция отображения окна с заданными параметрами, такими как высота и ширина. Средства библиотеки позволяют работать с примитивами, такими как линия, прямоугольник, круг и многоугольник, и со спрайтами, в которые можно загружать текстуры. С графическими объектами можно производить манипуляции, например, двигать их, менять размеры, угол, под которым пользователь будет видеть кадр. Функция-метод `create(VideoMode(windowWidth, windowHeight), string)`, относящаяся к классу `Window`, является основополагающей, так как используется при написании любого проекта, основанного на графической библиотеке SFML. Она открывает окно, в котором происходят все графические манипуляции.

События считываются с помощью метода `pollEvent (event)` класса `Event`. За работу с аудиофайлами отвечает заголовочный файл SFML/Audio.hpp, который позволяет работать с длинными музыкальными и короткими звуковыми файлами [5]. Из-за большого количества возможностей и методов, для написания игры значительная часть разработчиков выбирает SFML.

1.3.Функциональные требования к проекту

Перед созданием программы, каждый разработчик точно знает, какой проект он хочет получить. И порой этот продукт необходимо описать самым тщательным образом. Иными словами, нужно знать, какие требования заказчик предъявляет к продукту. Требования к программной системе часто классифицируются как функциональные, нефункциональные и требования предметной области.

Функциональные требования – это перечень сервисов, которые должна выполнять система, причём должно быть указано, как система реагирует на те или иные входные данные, как она ведёт себя в определённых ситуациях. В некоторых случаях указывается, что система не должна делать. Если функциональные требования оформлены как пользовательские, они, как правило, описывают системы в обобщенном виде. В противоположность этому функциональные требования, оформленные как системные, описывают систему максимально подробно, включая ее входные и выходные данные, исключения.

В данном курсовом проекте функциональные требования представляют собой следующий список:

1.Реализовать уникальный проект жанра «Action» с особенностями игр-платформеров. Максимально оптимизировать и отладить работу создаваемой программы.

2.В проекте должен быть полностью проработан интерфейс для удобной работы, прописана и реализована игровая логика персонажей, добавлены элементы игр жанра «RPG», такие как выпадение предметов после убийства, повышение уровня и, соответственно, улучшение навыков персонажа.

3.Должен быть описана история игрового мира и главного героя, все прохождение игры должно сопровождаться соответствующей музыкой и звуками.

4.Проект должен быть нацелен на охват максимально большой аудитории пользователей, с этим условием должны создаваться все текстуры и эффекты в игре.

Это не полный список, потому что требования к функционалу могут включать в себя, помимо прочего, подробное описание того, что в них войдет (собственно, данные), как они будут отображаться (интерфейс), а также показатели эффективности, демонстрирующие, работает ли функционал так, как задумано.

2.СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

Основной функцией компьютера является обработка информации. Программная обработка данных на компьютере реализуется следующим образом:

1.После запуска на выполнение программы, хранящейся во внешней долговременной памяти, она загружается в оперативную память.

2.Процессор последовательно считывает команды программы и выполняет их.

3.Необходимые для выполнения команды данные загружаются из внешней памяти в оперативную и над ними производятся необходимые операции. Данные, полученные в процессе выполнения команды, записываются процессором обратно в оперативную или внешнюю память.

4.В процессе выполнения программы процессор может запрашивать данные с устройств ввода информации и пересылать данные на устройства вывода информации.

Программная конфигурация ПК многоуровневая. Это связано с тем, что требования к программам, предназначенным для работы с устройствами, существенно отличаются от требований к программам, предназначенным для работы с людьми. Общий принцип такой: чем ниже уровень программ, тем больше они работают с устройствами и меньше с человеком. Этот принцип соблюдается во всей компьютерной технике от отдельного ПК до всемирной компьютерной сети Интернет.

В понятие структуры программы включается состав и описание связей всех модулей, которые реализуют самостоятельные функции программы и описание носителей вводимых и выводимых данных, а также данных, участвующих в обмене между отдельными подпрограммами [6].

Для разработки больших и сложных программ программисту необходимо овладеть специальными приемами получения рациональной структуры программы, которая обеспечивает почти двукратное сокращение объема программирования. Подчиненность модулей программы отражается в схеме, представленной в приложении А. Однако последняя не отражает порядок их вызова или функционирование программы.

2.1.Разделение программы на составные части

Основным принципом разделения программы был провозглашен функциональный. Разделение исходного текста программы на несколько файлов становится необходимым по многим причинам. Среди них основным является неудобство работы с большим количеством текста и использование отдельных модулей, решающих разные задачи. Разделение по последним позволяют компилировать каждый модуль отдельно, а затем объединить их все в процессе построения конечной программы. Процесс объединения отдельно скомпилированных модулей называется **компоновкой**.

Расчленение программы на подпрограммы производится по принципу от общего к частному, более детальному. Процесс составления функционального описания и составления схемы иерархии является итерационным, а выбор наилучшего варианта является многокритериальным. Расчленение должно обеспечивать удобный порядок ввода частей в эксплуатацию.

Во-первых, это уменьшает время компиляции. Большие программы требуют больше времени на компоновку. Пересобирать программу целиком каждый раз, когда была изменена одна функция, очень неудобно. Намного проще откомпилировать одну функцию, организованную в виде отдельного модуля.

Во-вторых, гораздо проще понимать, писать и отлаживать программу, состоящую из нескольких хорошо продуманных модулей, каждый из которых логически объединяет несколько связанных между собой функций. Крайне трудно разобраться в содержимом одного объемистого модуля, охватывающего все относящиеся к программе функции. Кроме того, по мере разрастания модуль становится все более запутанным и в результате его приходится неоднократно переписывать.

В данном курсовом проекте программный код разбит на несколько фрагментов. под **фрагментом** подразумевается не просто произвольный кусок кода, а функция, или группа логически связанных функций, или класс, или несколько тесно взаимодействующих классов.

Фрагментами данного проекта являются: «Главное меню» – содержащий методы для обработки действий пользователя и обеспечивающий правильное отображение интерфейса программы. «Главное меню» позволяет перейти к следующим трем фрагментам: «Выход» – соответственно закрытие программы, «Помощь» – предоставляющий пользователю информацию о правильном использовании программы, и основной фрагмент – «Начать игру». Последний загружает карту, текстуры, музыку и звуки с помощью описанных в нем методах. Далее все зависит от действий пользователя по игре. Требуется нажать определенную комбинацию клавиш и перейти в «Пауза», откуда сможет вернуться в «Главное меню» либо продолжить игру. В случае, когда пользователь использует функционал проекта согласно планам разработчика, он может запустить один из следующих фрагментов: «Наносить урон, убивать врага» и «Получать урон, смерть героя» описывающие главную особенность игры – уникальную логику противников и боевую систему проекта. Последний, при выполнении реализованного в нем условия `if(p.health <= 0) p.life=false;` перенаправляет пользователя снова в «Главное меню». Так же во время управления персонажем пользователь может пользоваться элементами игры-платформера, а именно функциями фрагмента «Собирать бонусы». Данный курсовой проект включает в себя несколько уровней, переход на которые осуществляется посредством выполнения миссий, за которые отвечает одноименный фрагмент программы. Прохождение всех

уровней, а соответственно и миссий, активирует «Завершение игры». В данном фрагменте описан интерфейс завершения прохождения игры.

Благодаря разделению проекта на части, разработчику может беспрепятственно дополнять свой проект новыми возможностями и, соответственно, тестировать их, не прибегая к работе во всем проекте сразу.

2.2.Сторонние программные компоненты

Модульность является одним из ключевых принципов разработки программного обеспечения с 1960-х годов. Применение этого принципа приносит в программирование много полезного. Модульность способствует эффективному использованию принципа разделения ответственностей, что ведёт к улучшению возможностей по созданию, многократному использованию, компоновке кода.

В наше время применение принципа модульности в проектировании ПО приняло новую форму, воплотившуюся в компонентах. Это – разработка, основанная на компонентах (Component Driven Development, CDD). Современные библиотеки и фреймворки для разработки пользовательских интерфейсов, такие как React, Vue и Angular, а также CDD-ориентированные инструменты наподобие Bit, позволяют создавать приложения, опираясь на модульные компоненты.

Компонент – это чётко очерченный независимый фрагмент интерфейса приложения. В качестве примеров компонентов можно привести такие сущности, как кнопки, слайдеры, окна для вывода сообщений чатов. Понимая особенности CDD и умея применять этот подход к разработке, можно использовать компоненты в качестве основы приложений [7].

Внешние компоненты – это сторонние библиотеки, которые подключаются к системе для расширения ее возможностей. Внешние компоненты используются для решения задач, которые сложно или невозможно реализовать на встроенном языке C++. Одним из таких компонентов, используемых в данном курсовом проекте является программа TiledMapEditor – один из редакторов тайловых карт. Он хорош тем, что карту, созданную в этом редакторе, состоящую из объектов, тайлов, их слоев, можно сохранить в XML-подобном файле .tmx и потом с помощью специальной библиотеки на C++ считать ее.

Применение CDD повышает эффективность разработки ПО. По словам Брэда Фроста, автора книги об атомарном дизайне, модульность и композиция – это важнейшие понятия в биологии, экономике, и во многих других областях. Модульность в применении к разработке программного обеспечения даёт скорость, надёжность, простоту разработки. Она способствует многократному использованию сущностей, улучшает тестируемость и расширяемость кода. Модульность даёт разработчику возможность применения композиции при создании сложных систем. Всё это очень хорошо влияет на процесс и на результаты разработки приложений.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была разработана 2D Игра «The Quest for the treasure unseen» Во время реализации данного проекта были использован язык программирования C++, сторонние библиотеки, такие как SFML, а также внешние программы, например, TiledMapEditor.

В программе удалось реализовать качественную детальную анимацию главного героя и его врагов. Добавлены различные виды оружия, а именно ближнего и дальнего боя, а также бонусы для протагониста, которые придают игре разнообразие и возможность перевернуть исход битвы в свою пользу. Описана логика противников, а также интересная механика – краткосрочная неуязвимость во время «скольжения», делающая геймплей игры более захватывающим и динамичным, реализованы динамические полоски здоровья всех персонажей, позволяющие контролировать ситуацию на поле боя и грамотно рассчитывать свои дальнейшие действия. Присутствует приятный и простой интерфейс, позволяющий даже неопытным пользователям пройти игру полностью, получив, соответственно, максимум положительных эмоций. Игра получилась интересной, быстрой и оптимизированной. Кроме того, способной конкурировать с другими проектами того же жанра. Однако существуют несколько пробелов, которые не были исправлены либо добавлены до момента предоставления данного проекта.

Звуки и музыка – это обязательная часть любой игры в современной игровой индустрии, однако в данном курсовом проекте они временно недоступны. Не реализован мультиплеер, целью которого является – сплочение игроков с различных уголков мира для достижения конечной цели – победить общего антагониста, а также позволить игрокам сражаться друг с другом, а, следовательно, создание турниров и таблицы лидеров, поощрение самых способных игроков внутриигровыми наградами. Необходимо описать логику врагов немного запутанней для ликвидации однотипности игры и повышения неожиданности. Все вышеуказанные и не только пробелы в игре были из-за недостаточно обширных знаний у автора курсового проекта и нехватке рабочего времени.

В дальнейшем планируется не только устранить данные проблемы, но и улучшить уже имеющиеся сильные стороны игры. Во-первых, развить интерфейс: добавить больше возможностей пользователю, например, изменение громкости звука и музыки, разрешения экрана, переключение языка или изменение уровня сложности. Во-вторых, увеличить количество уровней, добавить новых рядовых противников и боссов, реализовать систему выпадения вещей, ввести внутриигровую валюту, а исходя из последней, реализовать транзакции. Все вышеперечисленные изменения и обновления игры позволят увеличить количество пользователей и популярности данного проекта, а это, в свою очередь, позволит найти спонсоров для будущих проектов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1].Язык программирования C++ [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/C++. – Дата доступа 19.9.2019.

[2].SFML [Электронный ресурс]. – Режим доступа: https://www.sfml-dev.org – Дата доступа 30.10.2019.

[3].Языки программирования [Электронный ресурс]. – Режим доступа: http://sd-company.su/article/help_computers/programming_language – Дата доступа 5.10.2019.

[4].Технологии программирования [Электронный ресурс]. – Режим доступа: http://info-master.su/programming/profi/programming-technologies.php – Дата доступа 7.10.2019.

[5].Кроссплатформенная мультимедийная библиотека SFML [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/SFML. – Дата доступа 1.10.2019.

[6].Структура программного обеспечения ПК [Электронный ресурс]. – Режим доступа: https://studopedia.ru/4_4312_tema--struktura-programmnogo-obespecheniya-pk.html – Дата доступа 20.10.2019.

[7].Руководство по разработке на компонентах [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/ruvds/blog/461661/ – Дата доступа 11.10.2019.

[8].Теоретические основы алгоритмизации и программирования [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/4693947/ – Дата доступа 19.10.2019.

Луцик, Ю. А. Объектно-ориентированное программирование на языке C++: учеб. пособие по курсу «Объектно-ориентированное программирование» для студ. спец. «Вычислительные машины, системы и сети» всех форм обуч. / Ю. А. Луцик, А. М. Ковальчук, И. В. Лукьянова. – Минск: БГУИР, 2003. – 203 с.:ил.

Коптенок, Е. В. Использование средств библиотеки SFML для написания игровых проектов: справ. пособие / Коптенок Е. В., Храмченков Е. В., Храмченко В. Д. – СПб: Техника. Технологии. Инженерия, 2018. – с. 18-22.

ПРИЛОЖЕНИЕ А
(обязательное)

Схема структурная

ПРИЛОЖЕНИЕ Б
(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ В

(обязательное)

Диаграмма последовательностей

ПРИЛОЖЕНИЕ Г
(обязательное)

Схема программы

ПРИЛОЖЕНИЕ Д
(обязательное)

Схема программы

ПРИЛОЖЕНИЕ Ж
(обязательное)

Ведомость документов