МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование информационных систем**

**Тема:** «Объектно-ориентированное моделирование. ДИАГРАММЫ ПОВЕДЕНИЯ UML.»

**Цель:** Изучение методологии объектно-ориентированного моделирования средствами UML. Получение дополнительных навыков проектирования моделей информационной системы с применением возможностей UML диаграмм.

Выполнил:

Тышкевич Р.А. ,

3 курс 7 группа

Проверил:

Якунович А. В.

Минск 2023

**Введение**

Unified Modeling Language (UML) — унифицированный язык моделирования. Расшифруем: modeling подразумевает создание модели, описывающей объект. Unified (универсальный, единый) — подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. UML описывает объект в едином заданном синтаксисе, поэтому где бы вы не нарисовали диаграмму, ее правила будут понятны для всех, кто знаком с этим графическим языком — даже в другой стране.

Одна из задач UML — служить средством коммуникации внутри команды и при общении с заказчиком. Давайте рассмотрим возможные варианты использования диаграмм:

* Проектирование. UML-диаграммы помогут при моделировании архитектуры больших проектов, в которой можно собрать как крупные, так и более мелкие детали и нарисовать каркас (схему) приложения. По нему впоследствии будет строиться код.
* Реверс-инжиниринг — создание UML-модели из существующего кода приложения, обратное построение. Может применяться, например, на проектах поддержки, где есть написанный код, но документация неполная или отсутствует.
* Из моделей можно извлекать текстовую информацию и генерировать относительно удобочитаемые тексты — документировать. Текст и графика будут дополнять друг друга.

Все рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют самостоятельного материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Однако для создания конкретной физической системы необходимо, некоторым образом, реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Чтобы пояснить отличие логического и физического представлений, рассмотрим в общих чертах процесс разработки некоторой программной системы. Ее исходным логическим представлением могут служить структурные схемы алгоритмов и процедур, описания интерфейсов и концептуальные схемы баз данных. Однако для реализации этой системы необходимо разработать исходный текст программы на некотором языке программирования (C++, Pascal, Basic/VBA, Java). При этом уже в тексте программы предполагается такая организация программного кода, которая предполагает его разбиение на отдельные модули.

Тем не менее исходные тексты программы еще не являются окончательной реализацией проекта, хотя и служат фрагментом его физического представления. Очевидно, программная система может считаться реализованной в том случае, когда она будет способна выполнять функции своего целевого предназначения. А это возможно, только если программный код системы будет реализован в форме исполняемых модулей, библиотек классов и процедур, стандартных графических интерфейсов, файлах баз данных. Именно эти компоненты являются необходимыми элементами физического представления системы.

Таким образом, полный проект программной системы представляет собой совокупность моделей логического и физического представлений, которые должны быть согласованы между собой. В языке UML для физического представления моделей систем используются так называемые диаграммы реализации (implementation diagrams), которые включают в себя две отдельные канонические диаграммы: диаграмму компонентов и диаграмму развертывания. Особенности построения первой из них рассматриваются в этой главе, а второй - в следующей.

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма деятельности – мощный инструмент, который интенсивно используется при создании ИС.

В зависимости от, поставленной перед нами задачи мы создаем диаграмму деятельности, используя тот набор элементов, который необходим для отражения определенного уровня детализации.

Таким образом, диаграмма деятельности может применяться как для описания бизнес-процесса, так и функциональных требований к Системе.

Цель концептуального описания - показать целостную картину бизнес-процессов предметной области.

**Описание программных средств**

Draw.io — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

Инструмент работает с Google Диск, Google Workspace и Dropbox, глубоко интегрирован и удобен для работы с продуктами Confluence и Jira от Atlassian. Пользователи также могут работать с диаграммами в автономном режиме и сохранять их локально, используя настольное приложение для персональных компьютеров.

Инструмент позволяет создавать: графики, диаграммы, таблицы, презентации, блок-схемы, планы помещений, воронки продаж, ментальные карты, карты сайтов.

В данной лабораторной работе при создании модели не были использованы библиотеки.

**Описание практического задания**

На данной диаграмме деятельности мы можем наблюдать процесс обновления игровой сцены.

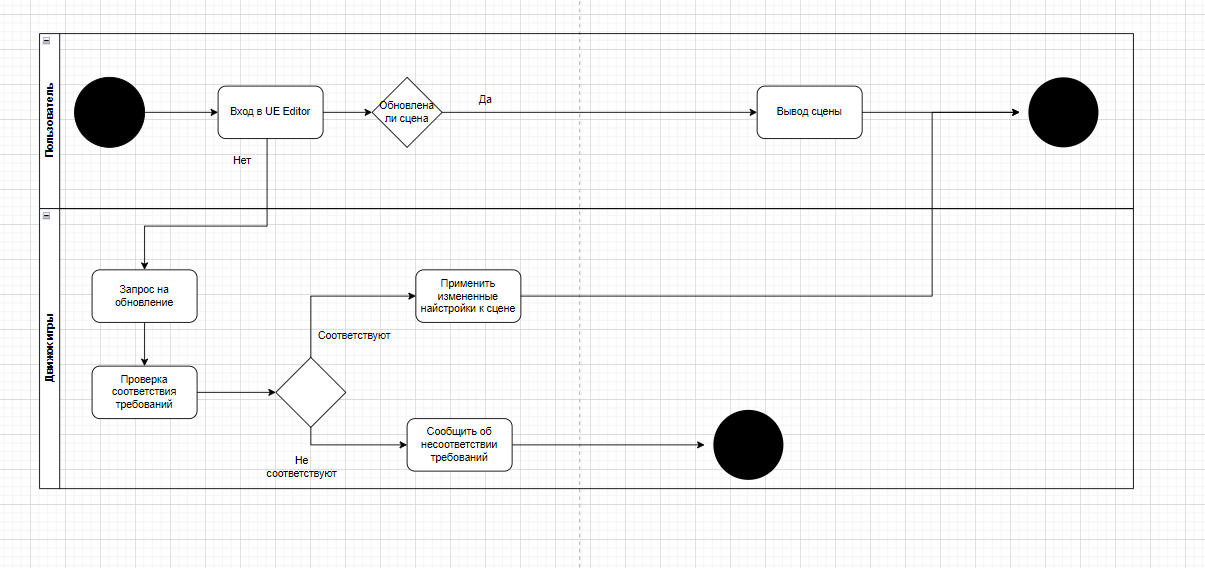


Рисунок 1 – UML Диаграмма деятельности

Основными элементами игрового процесса являются действия(actions): «Выбрать уровень», «Изменить настройки», «Запрос на обновление», «Проверка соответствия требований», «Применить измененные настройки к уровню», «Сообщить об несоответствии требований», «Обновить уровень». Также были использованы два блока решения, начальный и конечный узлы.

Операция «Проверка соответствия требований» необходима для проверки, занято ли игровое поле, выполняет ли игрок задание и т.д. при изменении настроек.

На данной диаграмме состояния продемонстрирован процесс просмотра игровой сцены.

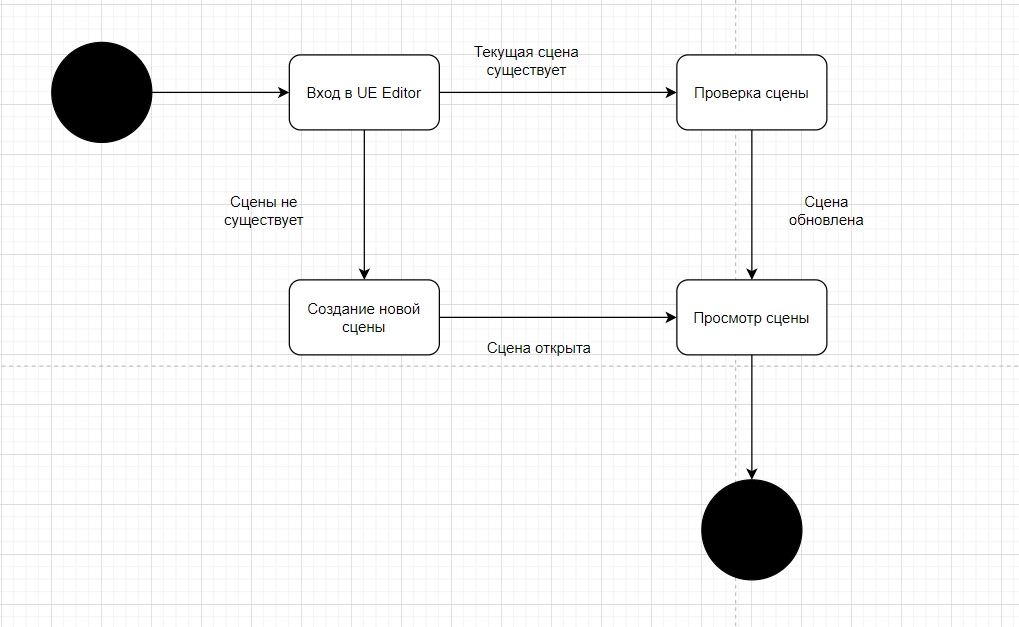


Рисунок 2 – UML Диаграмма состояния

Основными элементами игрового процесса являются следующие объекты: начальное псевдосостояние (initial pseudostate); Состояния (states) «Вход в игру», «Создание нового персонажа», «Авторизация», «Просмотр уровней»; Конечное состояние (final state). В качестве связей используются переходы, которые определяют:

1. Статус игрока в системе (зарегистрирован/не зарегистрирован);
2. Статус уровней.

**Вывод:** Я изучил методологию объектно-ориентированного моделирования средствами UML. Получил дополнительные навыки проектирования моделей информационной системы с применением возможностей UML диаграмм.