Федеральное государственное автономное
Образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Вычислительная техника

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О.В. Непомнящий
подпись инициалы, фамилия
«_____» _____ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Редактор уровней для игры "Картинки из геометрических фигур"

Руководитель _____ ст. преподаватель В.С. Васильев
подпись, дата

Выпускник _____ М.А. Баландин
подпись, дата

Нормконтролер _____ ст. преподаватель В.С. Васильев
подпись, дата

Красноярск 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Пазлы являются классической игрой, направленной на развитие логического и пространственного мышления у детей. Существует множество видов пазлов, но вне зависимости от вида пазла – цель их сбора заключается в получении единой картинки из отдельных частей.

Одной из популярных разновидностей пазлов является танграм – в ходе сбора которого ребенок составляет картину из разноцветных геометрических фигур – часто только многоугольников, но встречаются реализации с кругами и овалами [1, 2, 3]. В статье [4] предлагается аналогичный вид пазла для школьников младших классов. При этом, рисунки-задания группируются по темам. Нередко танграм совмещают с раскраской – тогда после сбора пазла от ребенка требуется правильно раскрасить его [3].

Игры такого типа нередко реализуются в виде мобильных приложений [5, 6, 7]. Для выпуска такого приложения необходимо наполнить его базу уровнями, которые могут группироваться по темам. Уровни описываются в определенном формате, создание таких файлов вручную – трудоемкая задача. **Цель работы** – создание инструментального средства, позволяющего разрабатывать и редактировать уровни для игры «Картинки из геометрических фигур», а также группировать их в тематические наборы.

В связи с тем, что приложение формирует уровни для игры, находящейся в стадии разработки, **задачами работы** являются:

- определение содержания уровня – типов фрагментов пазла;
- определение набора допустимых операций с элементами пазла;
- разработка формата файла, удобного для хранения данных уровня.

1 Спецификация требований

В ходе игры, ребенок составляет из предоставленного набора геометрических фигур заранее определённую картинку. Не имеет значение в какой области экрана будет собрана картинка, главное – правильно расположить фигуры друг относительно друга.

1.1 Анализ существующих аналогов игры

Таблица – Основные показатели приложения на Google Play [8]

Название Критерий	Танграм King	Танграм Мастер	Poly Shape
Наличие категорий	Да*	Да	Нет
Наличие овалов, кругов и линий	Нет	Нет	Нет
Возможность вращения	Да*	Да	Да
Взаимозаменяемые объекты	Нет	Нет	Нет
Оценка (количество звезд)	4,5	4,4	4,9
Количество скачиваний(тысяч)	1000	1000	1

Во всех рассмотренных приложениях игровое поле представляет собой фон и контур, в который нужно поместить фигуры так, чтобы они не перекрывали друг друга.

Картинка в «Танграм King» [5] всегда состоит из 7 фигур, все доступные фигуры располагаются вокруг контура собираемой картинки. Фигуры не взаимозаменяемы и должны занимать свои строго определённые места. В игре существуют категории, но они относятся не к тематике рисунков.

В приложении «Танграм Мастер» [6] присутствует деление по темам: люди, кошки и собаки, алфавит, животные, разное. Все фигуры на игровом поле можно вращать. Фигуры не взаимозаменяемы и должны занимать свои строго определённые места.

В «Poly Shape» [7] картинки состоят из разного количества фигур. Все фигуры можно вращать, но они не взаимозаменяемы. Отсутствует деление уровней на категории.

Таким образом, в редакторе необходимо реализовать возможность создания уровня для игры типа «Танграм», при этом он должен обеспечивать:

- добавление на рабочую область различных плоских фигур и линий;
- изменение свойств фигур: цвета, положения, угла поворота;
- задание точек привязки, задающих положение фигур относительно друг друга;
- возможность группировки уровней по категориям, добавления к ним дополнительной информации, например – тег уровня;
- экспорт информация об уровне в формат, удобный для обработки игрой.

В соответствии с описанными функциями, которые необходимо реализовать, с помощью языка моделирования UML [9] была разработана диаграмма прецедентов, представленная на рисунке 1.1.

Рисунок 1.1 – Диаграмма прецедентов для приложения «Редактор уровней»

1.2 Разработка требований для редактора уровней

Уровень представляет собой набор фигур, которые состоят из примитивов, таких как: овал, прямоугольник, треугольник, прямая. Каждая из фигур обладает набором свойств: цвет заливки, цвет контура, толщина

контура, точка отрисовки, точки привязки. Точки привязки нужны для проверки правильного положения фигур друг относительно друга.

Файл уровня должен содержать описание фигур, из которых состоит картинка, так же необходимо хранить номер уровня, для того чтобы в игре можно было задать последовательность их прохождения.

Необходимо иметь возможность формировать из набора файлов уровней файл группы уровней. Группа уровней – набор уровней одной тематики, в котором каждому уровню присвоен номер. Номер уровня уникален и не должен повторяться в пределах группы. Каждая группа принадлежит к какой-либо тематике (животные, растения и так далее).

1.2.1 Описание формата файла

После анализа требований к содержимому файла, для хранения данных уровня был выбран формат json.

Данные в файле должны описывать фигуры, которые содержат графические примитивы. Фигура хранит в себе примитивы, которые должны рассматриваться в игре, как один объект (например, глаз, состоящий из 2 окружностей; рот, который состоит из линии и нескольких одинаковых треугольников). Каждая фигура содержит в себе помимо стандартных для неё полей массив точек привязки, каждая точка привязки описывается идентификатором (уникальное поле), двумя целыми числами, которые описывают положение точки в фигуре и не привязаны к общей системе координат сцены и идентификатором точки другой фигуры, с которой она должна совпадать. Это позволит правильно определять расположение фигур друг относительно друга в игре. Каждый уровень так же имеет номер, это поле изначально пустое, пользователь назначает номер уровня при экспорте.

Файл группы уровней так же имеет формат json, файл содержит в себе последовательность файлов уровней, уровни располагаются по возрастанию их номера, который задал пользователь в окне экспорта.

Приложение должно использовать файловую систему компьютера для хранения файлов-уровней и файлов групп уровней.

На рисунке 1.2 изображено содержимое файла-уровня с одной фигурой, внутри которой содержится единственный примитив – овал.

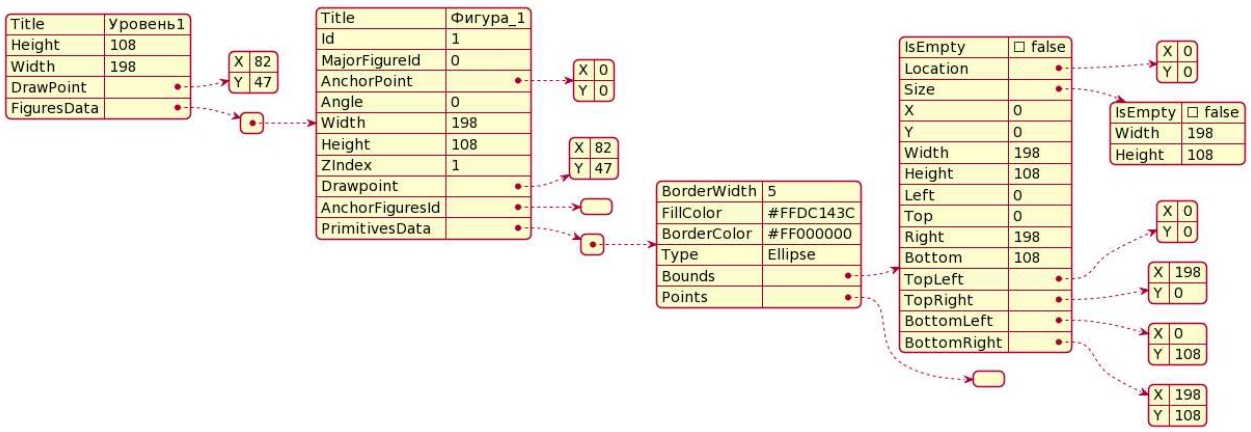

Рисунок 1.2 – Данные файла-уровня

1.2.2 Интерфейс программы

На рисунках 1.3 и 1.4 представлены макеты главного окна и окна экспорта соответственно. На рисунке 1.5 представлена диаграмма переходов между окнами программы.

На главном экране находится панель инструментов, пользователь может выбрать фигуру, которую хочет добавить на рабочую область или указатель, для выбора конкретной фигуры, изменения её свойств и положения; панель свойств, где отображаются свойства (цвет, ширина границ) выбранной фигуры, кнопка добавления точки привязки; меню, с помощью которого пользователь может взаимодействовать с файлом (открыть/создать, сохранить) или открыть окно экспорта; таблица элементов, в которой элементы могут быть объединены в один слой (т.к. некоторые элементы рисунка должны быть представлены в игре одним объектом, например, глаз, состоящий из 2 окружностей).

В окне экспорта находится рабочая область, в которую пользователь может добавлять файлы уровней перетаскивая их в область окна, панель свойств, в котором пользователь может задать тег и номер для выбранного уровня. Нажав на кнопку сохранить, пользователь инициализирует создание файла сета, который будет содержать все добавленные уровни с установленной для них информацией (номер уровня, тег). Номер уровня – уникальное свойство, которое не может повторяться в пределах сета. Тег – устанавливается сразу для всех уровней, если пользователь изменит тег одного из уровней теги основных уровней изменятся автоматически.

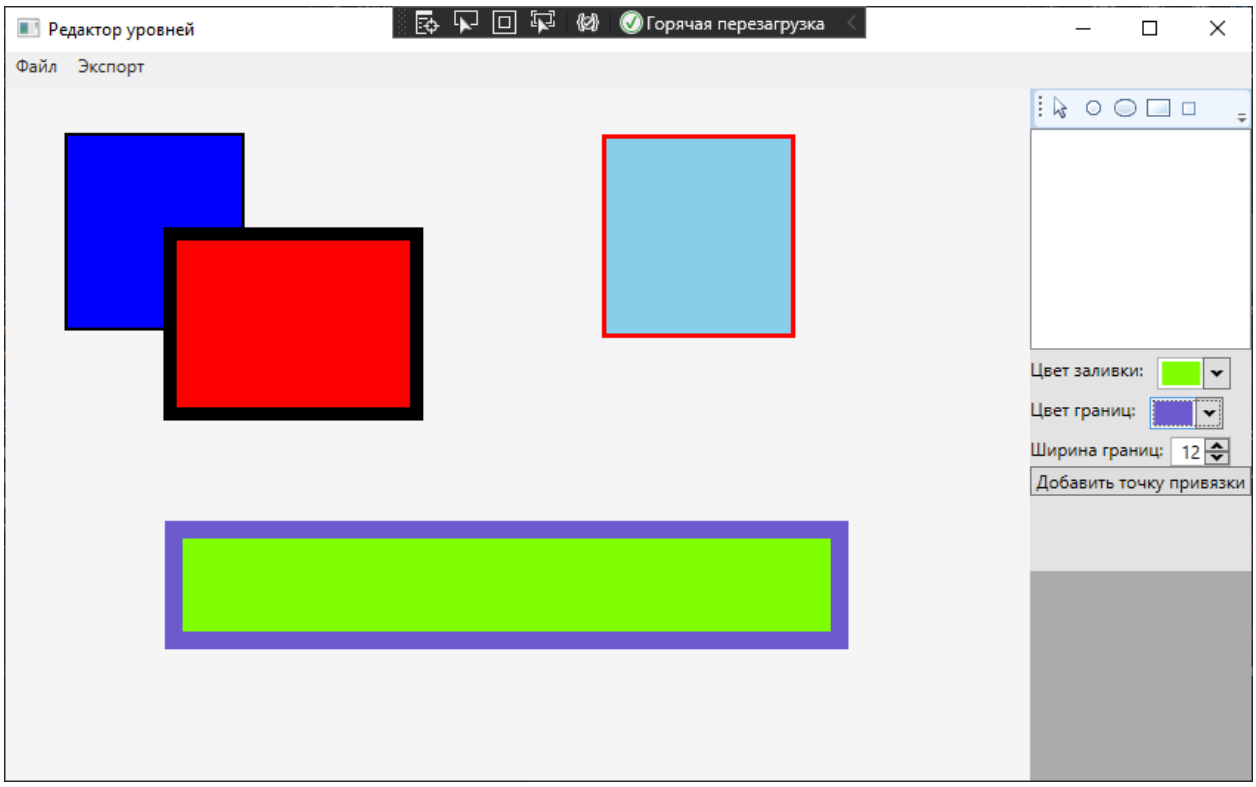

Рисунок 1.3 – Макет интерфейса главного окна приложения «Редактор уровней»

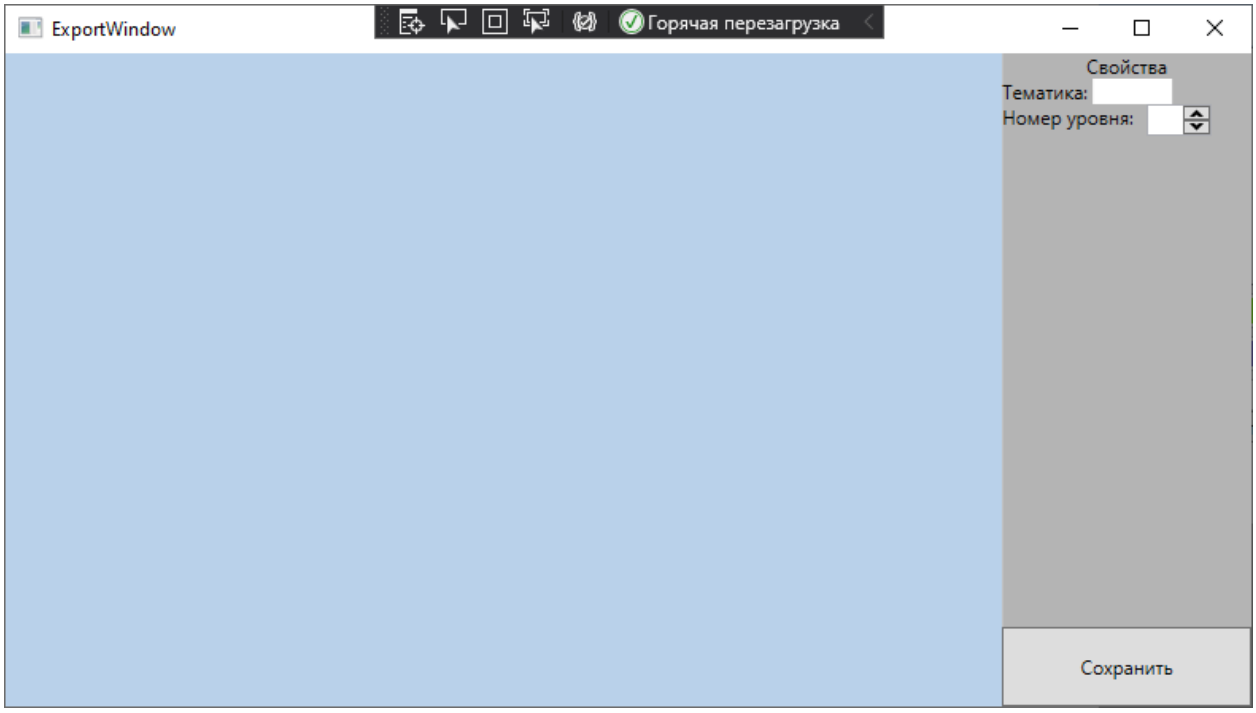

Рисунок 1.4 – Макет интерфейса окна экспорта приложения «Редактор уровней»

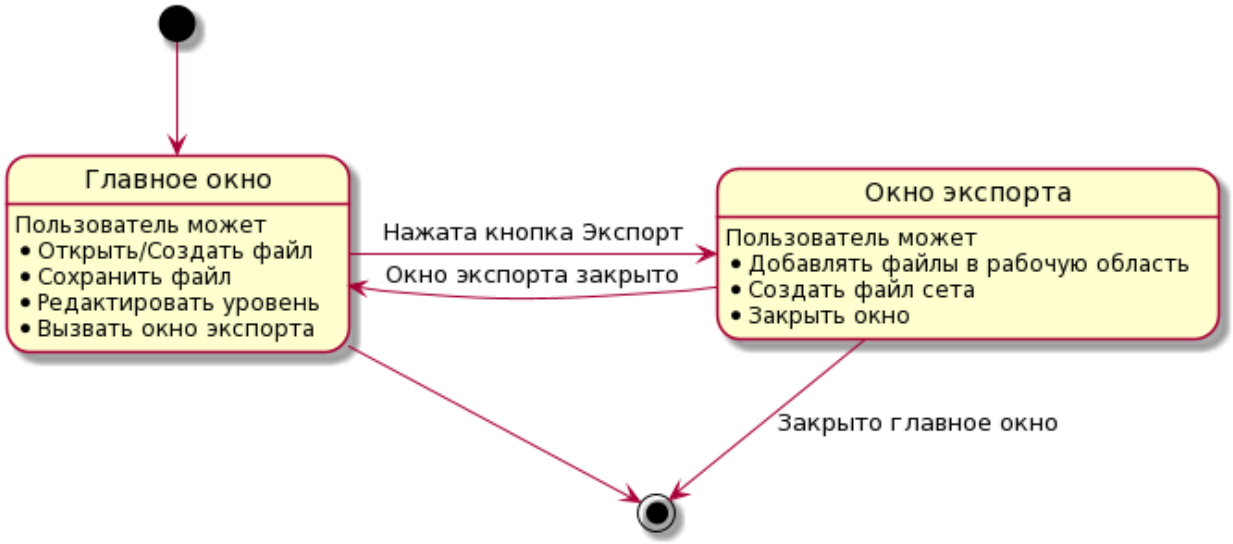

Рисунок 1.5 – Диаграмма потока экранов приложения «Редактор уровней»

1.2.3 Функциональные требования к программе

Функциональные требования выражены в виде диаграммы прецедентов, представленной на рисунке 1.1. Для каждого прецедента было сформировано текстовое описание, для наиболее сложных прецедентов были разработаны диаграммы пригодности [10].

Прецедент – Создать/Открыть файл

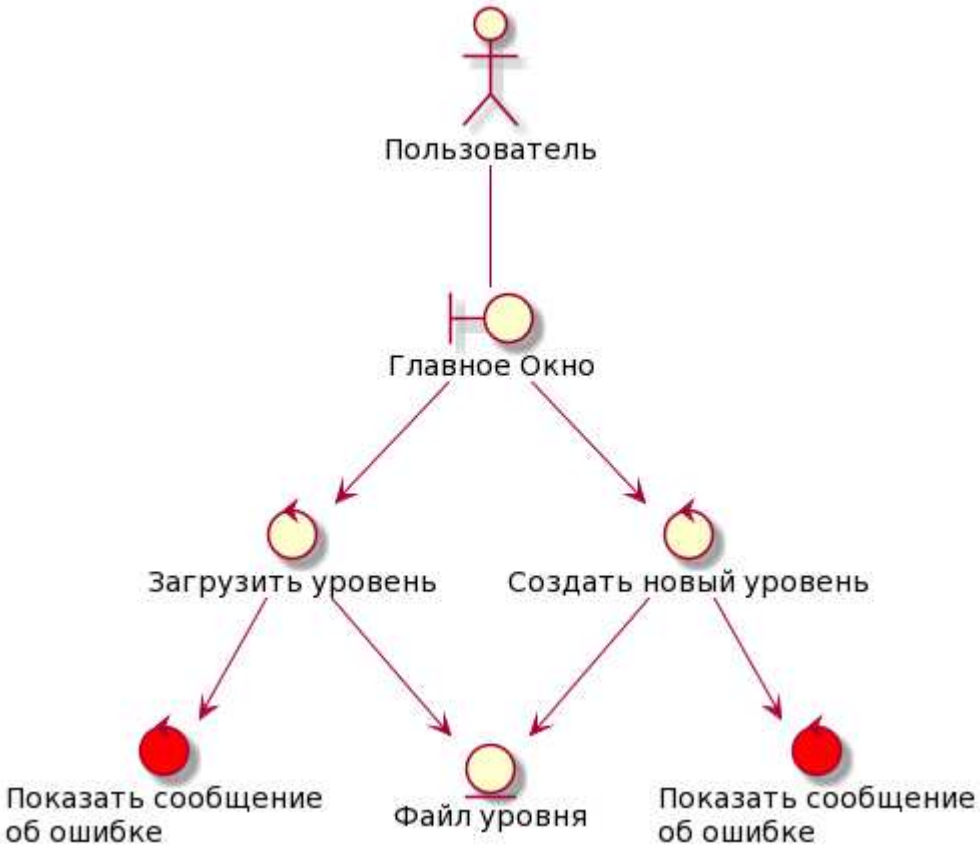

Рисунок 1.6 – Диаграмма пригодности для прецедента «Создать/Открыть файл»

Предусловие: Пользователь находится в главном окне программы.

Цель: создать/открыть файл рисунка-уровня для дальнейшей работы.

Основная последовательность:

1) Пользователь открывает меню действий с файлом;
2) Пользователь нажимает кнопку «Открыть» или «Создать»;
3) В появившимся диалоге пользователь выбирает файл рисунка-уровня или вводит его название;
4) Если вызвано создание, пользователь выбирает место хранения и имя файла;
5) Если выбрано открытие файла, то будет произведена загрузка данных;
6) Соответствующие данные отобразятся на рабочей области.

Постусловие: Файл успешно открыт в программе.

Условия ввода и действие альтернативных сценариев:

Условие 1: Попытка создать файл в каталоге с уже существующим файлом.

1) Программа выдаст ошибку о существовании файла.

Условие 2: Некорректный формат файла.

1) Программа выдаст ошибку об открытии файла.

Прецедент – Сохранить файл

Предусловие: Пользователь находится в главном окне программы.

Цель: Сохранить файл рисунка-уровня.

Основная последовательность:

1) Пользователь открывает меню действий с файлом;
2) Пользователь нажимает кнопку «Сохранить;
3) Файл сохраняется в директории по умолчанию;

Постусловие: Файл успешно сохранён.

Условия ввода и действие альтернативных сценариев:

Условие 1: Попытка сохранить файл в каталоге с уже существующим файлом.

1) Программа выдаст ошибку о существовании файла;
2) Пользователь может отменить действие или заменить существующий файл.

Условие 2: Некорректный формат файла.

1) Программа выдаст ошибку о сохранении файла.

Прецедент – редактирование уровня

Предусловие: Пользователь находится в главном окне программы.

Цель: Редактирование рисунка-уровня.

Основная последовательность:

1) Пользователь добавляет фигуру на рабочую область;
2) Пользователь изменяет свойства выбранной фигуры;
3) Пользователь задаёт точки привязки.

Постусловие: Данные рисунка-уровня изменены и сохранены.

Прецедент – Добавить фигуру

Предусловие: Пользователь находится в главном окне программы.

Цель: Добавление фигуры в рабочую область.

Основная последовательность:

1) Пользователь выбирает на панели инструментов фигуру, которую хочет добавить;
2) Пользователь выбирает точку на рабочей области и нажимает на неё;
3) Пользователь рисует фигуру, передвигая курсор на рабочей области не отпуская кнопку мыши.
4) Пользователь отпускает кнопку мыши.

Постусловие: Фигура добавлена в рабочую область.

Условия ввода и действие альтернативных сценариев:

Условие 1: Текущий инструмент – указатель.

1) Фигура не будет создана.

Прецедент – Изменить свойства фигуры

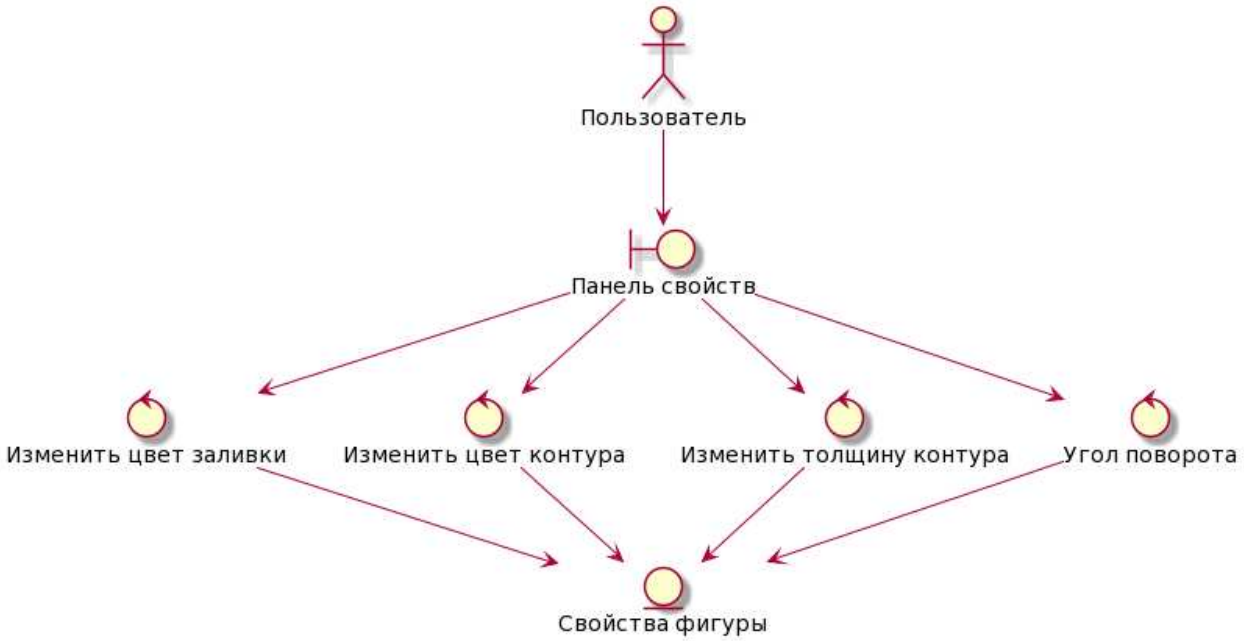

Рисунок 1.7 – Диаграмма пригодности для прецедента «Изменить свойства фигуры»

Предусловие: Пользователь выбрал фигуру в рабочей области**.**

Цель: Изменение свойств фигуры.

Основная последовательность:

1) Пользователь выбирает на панели свойств то свойство, которое хочет изменить;
2) Пользователь изменяет выбранное свойство.

Постусловие: Свойства выбранной фигуры изменены.

Прецедент – Задать точки привязки

Предусловие: Пользователь выбрал две фигуры, которые хочет связать.

Цель: Связать положение двух выбранных фигур друг относительно друга.

Основная последовательность:

1) Пользователь выбирает фигуру, которую хочет связать с другой фигурой;
2) Пользователь нажимает на кнопку «Связать с»;
3) Пользователь выбирает вторую фигуру, к которой будет привязана первая.

Постусловие: Две фигуры связаны(образована пара «Главный-зависимый»).

Условия ввода и действие альтернативных сценариев:

Условие 1: Пользователь пытается повторно связать фигуру.

1) Программа выведет диалоговое окно с информацией о том, что фигура уже связана.

Условие 2: Пользователь пытается создать связь, при которой две фигуры будут главными друг для друга.

1) Программа выведет диалоговое окно с информацией о том, что действие невозможно выполнить.

Прецедент – Объединить фигуры

Предусловие: Пользователь находится в главном окне программы.

Цель: Объединить две фигуры в одну.

Основная последовательность:

1) Пользователем выбирает первую фигуру нажатием на неё;
2) Пользователь нажимает кнопку «Объединить с»;
3) Пользователь выбирает вторую фигуру

Постусловие: Две фигуры объединены в одну.

Условия ввода и действие альтернативных сценариев:

Условие 1: При выборе второй фигуры пользователь нажал на пустое место рабочей области.

1) Программа выведет диалоговое окно с информацией о том, что действие невозможно.

Условие 2: Пользователь при выборе второй фигуры пользователь нажимает на первую.

1) Программа выведет диалоговое окно с информацией о том, что фигуру невозможно объединить саму с собой.

Прецедент – Загрузить набор уровней

Предусловие: Пользователь находится в окне экспорта.

Цель: Загрузить файл группы уровней.

Основная последовательность:

1) Пользователь переносит файл набора уровней в окно экспорта.

Постусловие: Уровни хранящиеся в файле набора уровней отображаются в окне экспорта в виде списка.

Прецедент – Изменить набор уровней

Предусловие: Пользователь находится в окне экспорта.

Основная последовательность:

1) Пользователь добавляет в рабочую область окна необходимые файлы уровней;
2) Пользователь удаляет из списка уровни, которые не должны содержаться в наборе;
3) Пользователь изменяет нумерацию уровней меняя элементы списка местами.

Постусловие: Набор уровней изменён.

Прецедент – Сохранение набора уровней

Предусловие: Пользователь сформировал список уровней, из которых необходимо сформировать набор.

Основная последовательность:

1) Пользователь сохраняет файл набора уровней.

Постусловие: Файл набора уровней сохранён.

1.3 Выводы по главе

Сформулированы требования к разрабатываемому приложению в виде макетов интерфейса, диаграмм прецедентов и их текстового описания, описан формат файла уровня и группы уровней.

2 Проектирование

Прецеденты описывают поведение пользователя в системе. Для визуализации взаимодействия компонентов между собой, а также с пользователем используются диаграммы последовательностей [11], отражающие динамическую модель системы. Результатом проектирования является статическая модель системы – диаграммы классов [12], на основе которых может быть выполнена генерация кода.

Приложение состоит из нескольких экранов, но практически вся логика относится к главному окну программы, в котором находится область для рисования.

В главном окне так же расположены: панель инструментов; меню, с помощью которого пользователь может взаимодействовать с файлом (открыть, сохранить) или открыть окно экспорта; окно свойств фигур; таблица элементов, в которой отображаются фигуры, их названия и лежащие в них примитивы.

2.1 Диаграммы последовательности

Значительная часть логики приложения связана с созданием объединением и привязкой фигур. В разделе приведены диаграммы последовательностей, для наиболее значимых прецедентов, связанных со взаимодействием различных компонентов системы.

На рисунке 2.1 изображена диаграмма последовательности для варианта использования «Добавить фигуру», наглядно представляющая процесс взаимодействия пользователя с программой для создания фигуры на рабочей области.

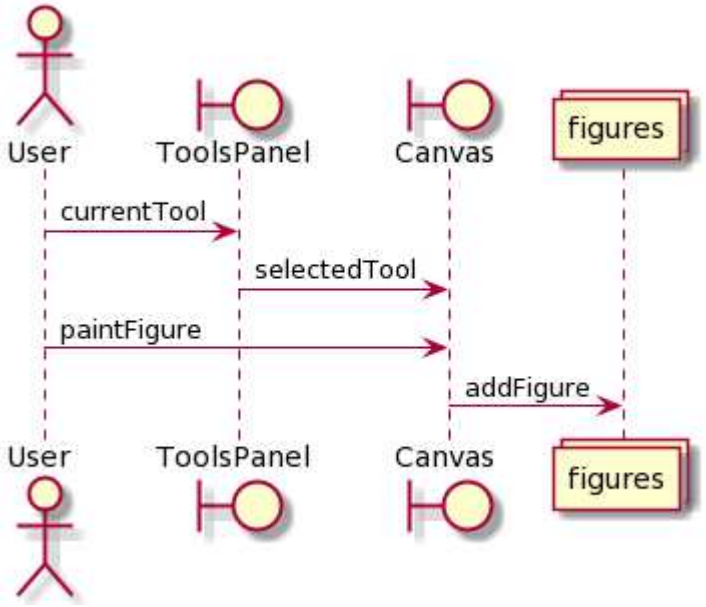

Рисунок 2.1 – Диаграмма последовательности «Добавить фигуру»

На рисунке 2.2 изображена диаграмма последовательности для варианта использования «Объединить фигуры», представляющая процесс объединение двух фигур в одну.

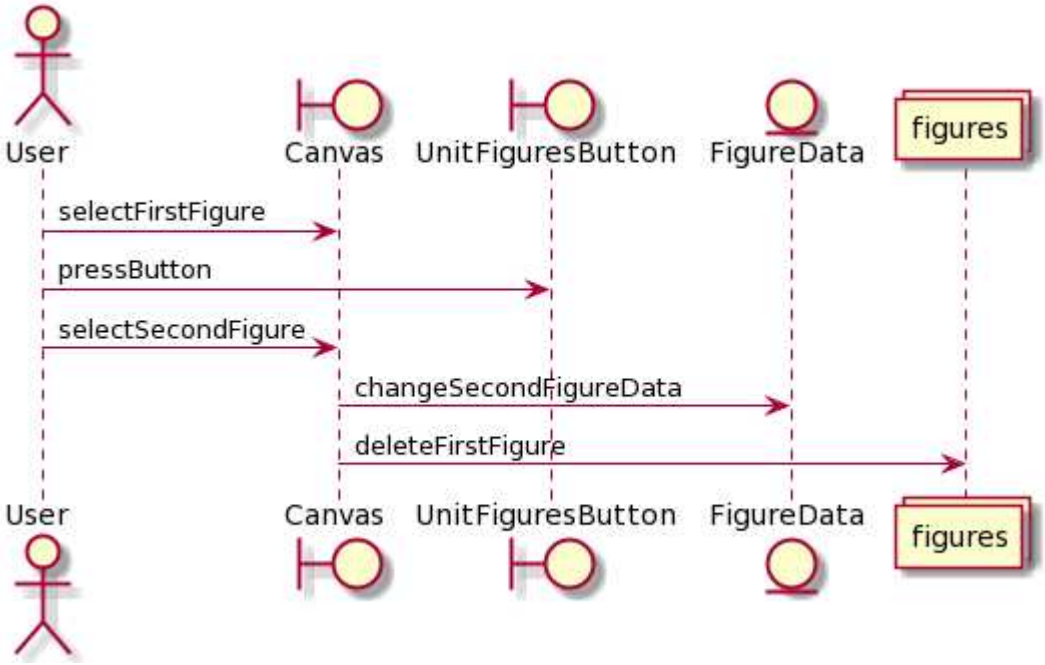

Рисунок 2.2 – Диаграмма последовательности «Объединить фигуры»

На рисунке 2.3 изображена диаграмма последовательности для варианта использования «Изменить свойства фигуры», представляющая процесс изменения свойств фигуры или её составляющих.

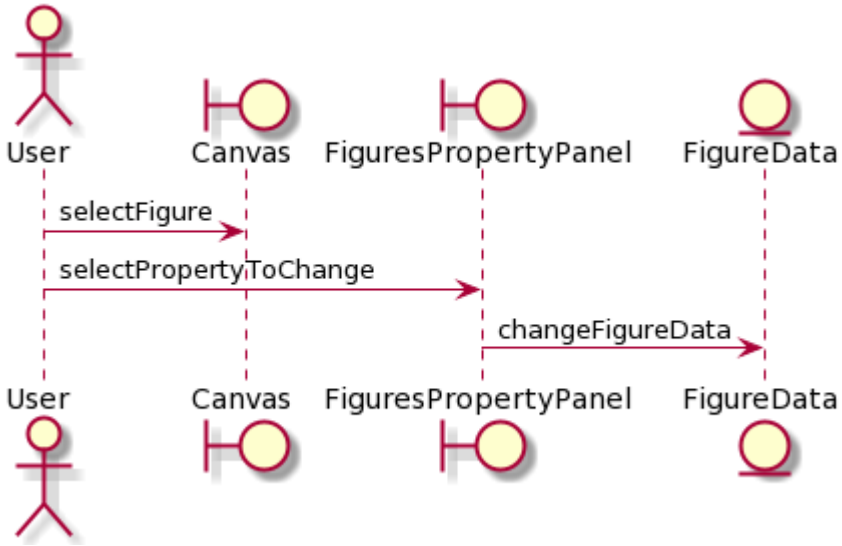

Рисунок 2.3 – Диаграмма последовательности «Изменить свойства фигуры»

На рисунке 2.4 изображена диаграмма последовательности для варианта использования «Открыть файл».

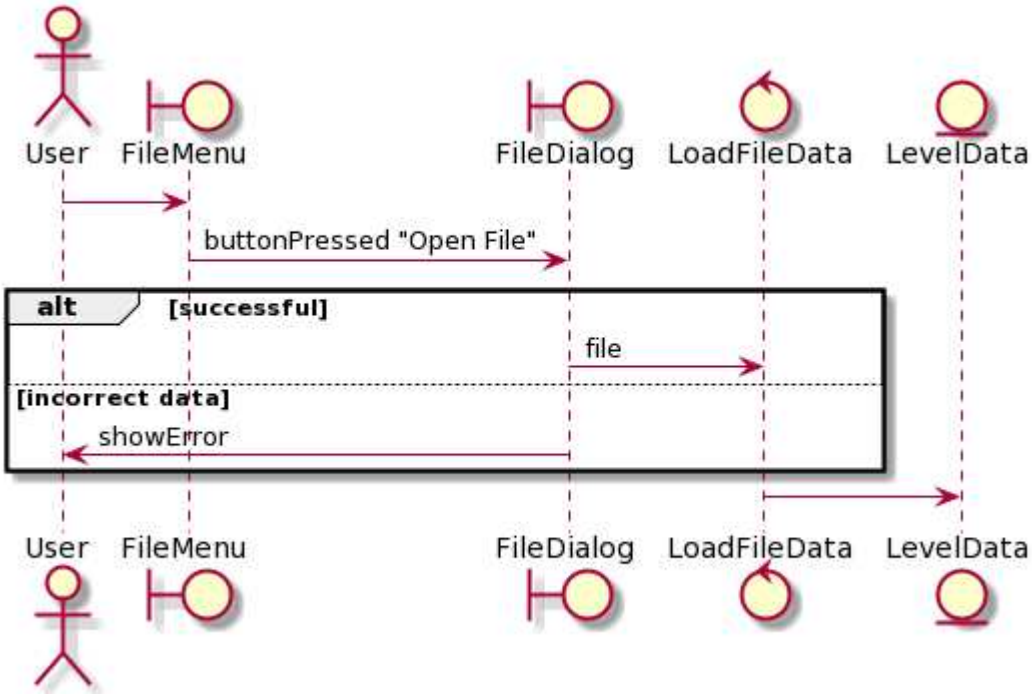

Рисунок 2.4 – Диаграмма последовательности «Открыть файл»

2.2 Основные алгоритмы программы

2.2.1 Объединение фигур

Фигура содержит в себе массив из одного или более примитивов, при объединении двух фигур, примитивы первой фигуры добавляются в массив примитивов второй фигуры, причём примитивы в массиве расположены в таком порядке, что примитив с индексом i+1 располагается поверх

примитива с индексом i. Результатом этой операции является фигура, которая содержит в себе все примитивы объединяемых фигур.

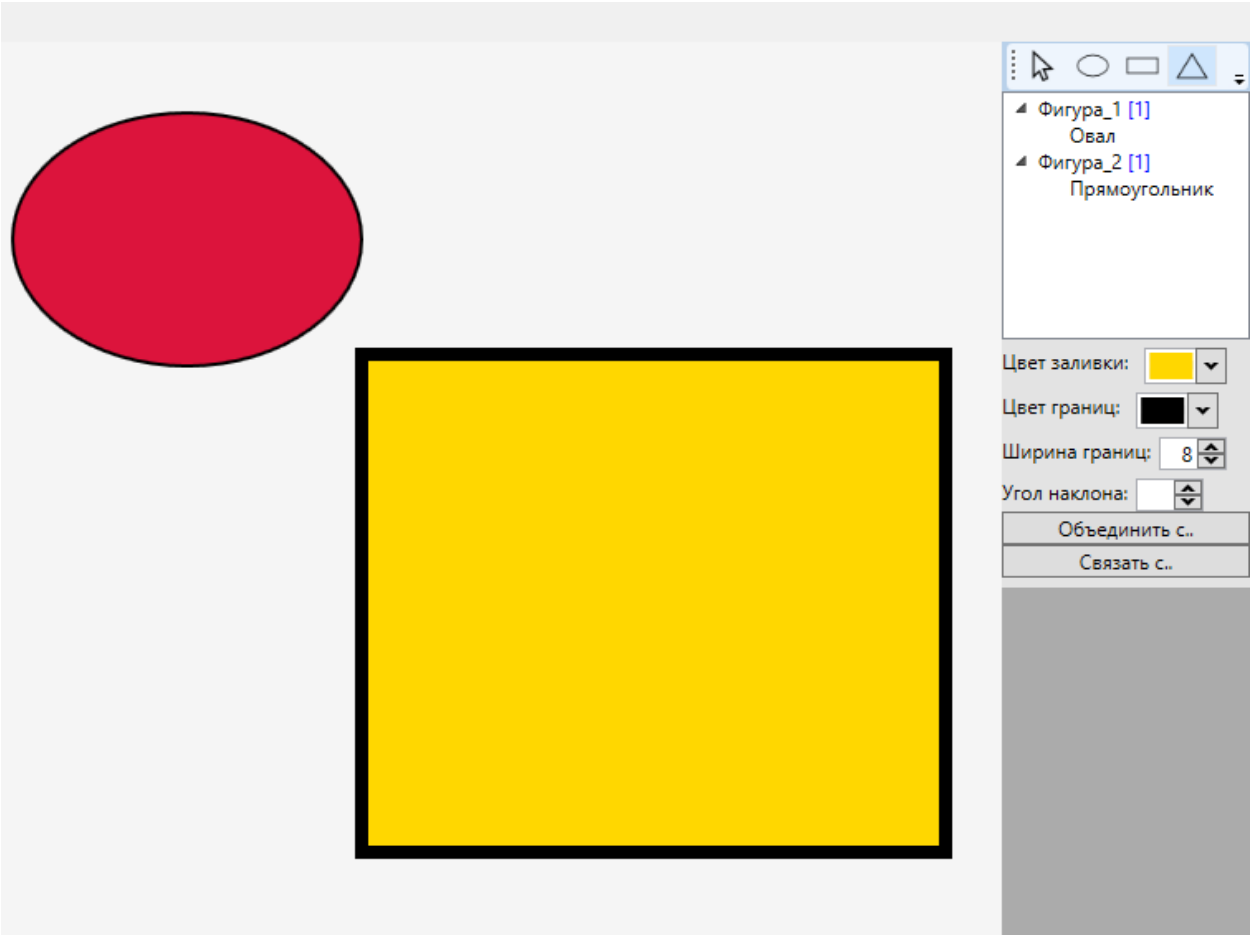

Рисунок 2.5 – Рабочая область с двумя отдельными фигурами

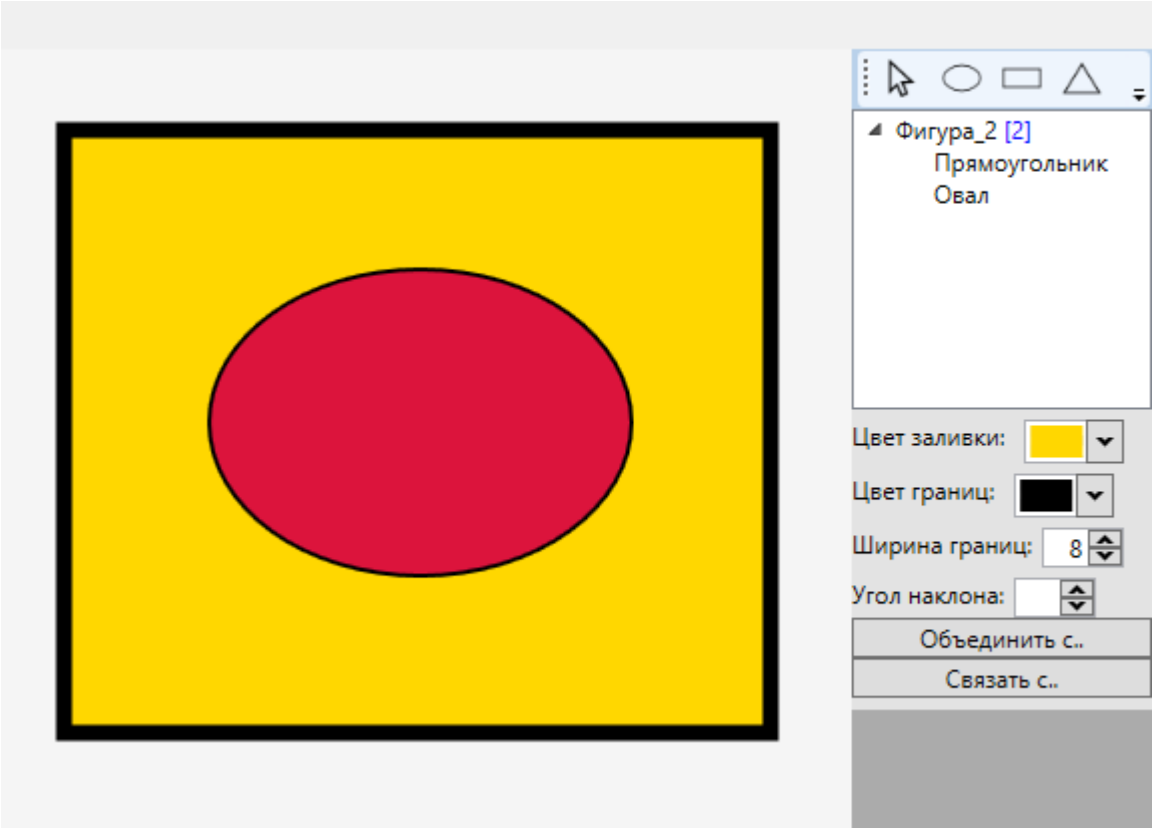

Рисунок 2.6 – Рабочая область с новой фигурой, которая создана при объединении двух отдельных фигур

2.2.2 Алгоритм привязки фигур друг к другу

Связывание двух фигур образует пару из главной и зависимой фигуры. У каждой фигуры может быть сколько угодно зависимых, но только одна главная, при этом фигуры не могут быть главными друг для друга. Каждая фигура имеет поле, хранящее идентификатор главной фигуры, массив, содержащий идентификаторы зависимых фигур, а также точку привязки.

Точка привязки представляет собой 2 значения: разница по оси X и разница по оси Y, относительно координаты точки отрисовки главной фигуры. Чтобы проверить правильно ли фигуры расположены друг относительно друга необходимо прибавить к координате точки отрисовки зависимой фигуры разницы по осям X и Y соответственно, если при этом координаты совпадут с координатами точки отрисовки главной фигуры, то можно считать, что фигуры расположены правильно.

Идентификаторы фигур начинаются с единицы, поэтому если поле, хранящее идентификатор главной фигуры, имеет значение 0, то считается что фигура не привязана.

2.3 Диаграммы классов

Конечный рисунок-уровень представляет собой набор фигур, в которых содержится один или более примитивов, примитив не может существовать отдельно от фигуры.

Фигуру можно перемещать, вращать, взаимодействовать с её отдельными примитивами (изменять цвет заливки, ширину и цвет границ), объединять или связывать с другими фигурами.

Фигура умеет перестраивать сама себя, если в неё добавляют или удаляют из неё примитив. Так же для классов Figure и Primitive реализован интерфейс IClonable для возможности использовать операции копирования и вставки на рабочей области.

Каждая фигура имеет уникальный Id, который не повторяется в пределах одного уровня. Также фигура хранит Id главной для себя фигуры (MajorFigureId), разницу по координатам X и Y до её точки отрисовки(AnchorPoint) и массив Id зависимых фигур(AnchorFiguresId).

Класс Primitive содержит свойство типа GeometryDrawing внутри которого может хранится любая простая фигура (прямоугольник, линия или массив линий, окружность), однако, чтобы корректно отображать вложенные в фигуру примитивы, Primitive также содержит свойство Type.

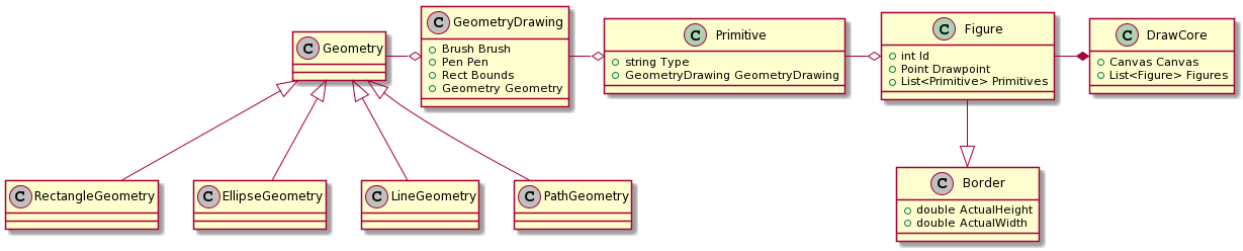

Рисунок 2.7 – Диаграмма классов модуля рисования

Для формирования файла уровня, а также для загрузки данных уровня написан статический класс Parser.

- Класс LevelData кроме полей для названия уровня и массива фигур содержит свойство DrawPoint, которое хранит в себе верхний левый угол ограничивающего прямоугольника для всей картинки. Вместе со свойствами Height и Width можно полностью описать этот прямоугольник. Это необходимо для того, чтобы в игре можно было правильно масштабировать фигуры.
- Класс FigureData содержит свойства, полностью описывающие фигуру.
- Класс PrimitiveData содержит свойство Bounds типа Rect, Rect – структура, описывающая ограничивающий прямоугольник. Все

примитивы, хранящиеся в фигуре, имеют координаты относительно фигуры, а не глобальной области.

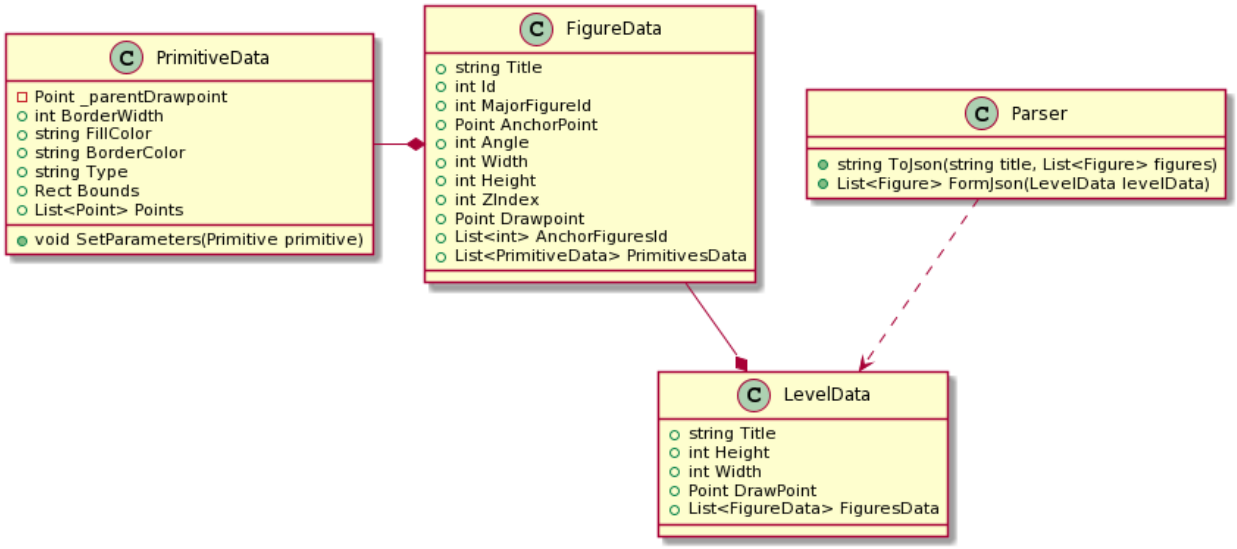

Рисунок 2.8 – Диаграмма классов парсера данных

2.4 Формат файла

В соответствии с требованиями к формату файла, описанными в пункте 1.2.1 файл уровня содержит название и тег уровня, координаты начала рисунка, его ширину и высоту (для скалирования рисунка в игре), массив фигур.

Каждая фигура имеет название, точку отрисовки (верхний левый угол ограничивающего его прямоугольника), ширину и высоту, Z индекс, уникальный идентификатор, идентификатор главной фигуры, точку привязки, массив идентификаторов зависимых фигур, массив примитивов.

Каждый примитив содержит информацию о своём типе (линия, прямоугольник и т.д.), цвете заливки, цвете и ширине границ, также примитив хранит структуру, которая описывает геометрию примитива внутри фигуры и массив точек, который используется только в примитивах типа многоугольник и ломаная.

Данная структура файла отображена на диаграмме классов парсера данных, представленного на рисунке 2.8.

На рисунке 2.9 представлено содержимое выходного файла-уровня, который содержит две фигуры, каждая из которых состоит из одного примитива. Фигура с именем «Зависимая» привязана к фигуре с именем «Главная».

Primer.json

Схема: <Схема не выбрана>


```
{
  "Tag": "ВКР",
  "Title": "Пример",
  "Height": 76,
  "Width": 233,
  "ModificationDate": "2021-06-05T15:10:29.7496475+07:00",
  "DrawPoint": {
    "X": 48,
    "Y": 28
  },
  "FiguresData": [
    {
      "Title": "Главная",
      "Id": 1,
      "MajorFigureId": 0,
      "AnchorPoint": {
        "X": 0,
        "Y": 0
      },
      "Width": 89,
      "Height": 75,
      "ZIndex": 0,
      "Drawpoint": {
        "X": 48,
        "Y": 28
      },
      "AnchorFiguresId": [ 2 ],
      "PrimitivesData": ...
    },
    {
      "Title": "Зависимая",
      "Id": 2,
      "MajorFigureId": 1,
      "AnchorPoint": {
        "X": -154,
        "Y": -1
      },
      "Width": 78,
      "Height": 70,
      "ZIndex": 1,
      "Drawpoint": {
        "X": 202,
        "Y": 29
      },
      "AnchorFiguresId": [],
      "PrimitivesData": ...
    }
  ]
}
```


Рисунок 2.9 – Пример выходного файла

2.5 Выводы по главе

На основе прецедентов разработаны диаграммы последовательностей, которые корректно визуализируют текстовые описания прецедентов.

На основе диаграмм последовательностей разработаны диаграммы классов, на основе которых может быть выполнена генерация кода.

Описаны основные алгоритмы программы.

3 Реализация, тестирование и документация

3.1 Реализация

3.1.1 Используемые инструменты

При реализации программы использовалась система Windows Presentation Foundation (WPF) [13], которая предоставляет средства для создания визуального интерфейса, элементы управления, привязку данных, двухмерную и трёхмерную графику, документы, текст, мультимедиа и оформление, и платформа .NET5 [14].

3.1.2 База данных

Игра «Картинки из геометрических фигур» использует удалённую базу данных MongoDB для хранения файлов-уровней. Для возможности взаимодействия с этой базой данных (добавления и обновления уровней) был написан класс LevelRepository. Класс содержит метод, который позволяет добавить уровень в базу данных. Для работы с базой данных используется библиотека MongoDB.Driver, которая предоставляет методы для взаимодействия с удалённой базой данных.

При экспорте файла-уровня осуществляется проверка на наличие этого уровня в базе данных. Производится попытка обновить уровень, который имеет тег и имя соответствующие экспортируемому файлу, если результат этой операции будет неудачным, это будет означать, что такого уровня её нет в базе и он будет добавлен. При этом будет обновлён файл конфигурации, который содержит дату последнего обновления базы данных. Все файлы в базе данных представляют собой json файлы.

3.1.3 Интерфейс и инструменты приложения

На основе макетов был разработан понятный интерфейс, приведённый на рисунке 3.1. Были добавлены дополнительные инструменты для операций с фигурами, такие как: на передний/задний план, отсоединить примитив, отвязать фигуры, пересчитать точки привязки. Добавлена возможность рисования многоугольников и ломаных.

Все неподписанные кнопки приложения имеют всплывающую подсказку, которая появляется при наведении курсора на соответствующую кнопку.

В нижней части главного окна приложения находится строка состояния, которая отображает текущие координаты курсора на холсте, размер холста, текущую операцию (перетаскивание, объединение, рисование)

и подсказку. Например, для операции рисования прямоугольника подсказка будет выглядеть так: «Нажмите и удерживайте ЛКМ, чтобы начать рисовать».

В правой части окна расположен список свойств примитива и фигуры. Для выбранной фигуры отображаются координаты начальной точки (левого верхнего угла ограничивающего её прямоугольника), размеры, Z индекс, имя, идентификатор главной фигуры и количество содержащихся в ней примитивов.

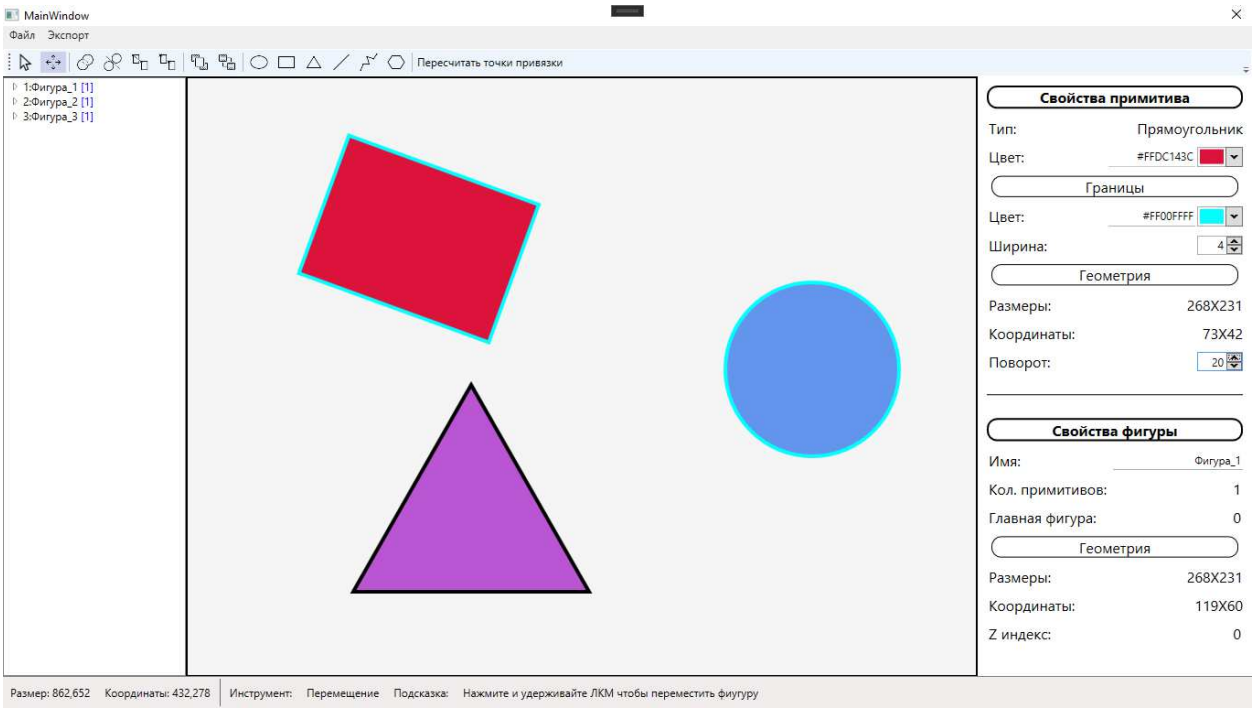

Рисунок 3.1 – Главное окно приложения

3.1.4 Привязка данных

WPF предоставляет механизм привязки данных [15], в приложении привязка используется для отображения свойств фигур и примитивов. Для возможности установки соединения между интерфейсом и данными классы Figure и Primitive имплементируют интерфейс INotifyPropertyChanged. Реализация интерфейса заключается в создании классом события изменения его свойства. Данные события содержат имя изменяемого свойства, его старое и новое значение. После реализации интерфейса на панели свойств данные о фигуре и примитиве, такие как размеры, координаты и т.п. отображаются в реальном времени.

3.1.5 Экспорт файлов

Приложение позволяет экспортировать только что созданный уровень в базу данных, для этого необходимо в верхнем меню нажать на кнопку «Экспорт», выбрать в открывшемся меню «Быстрый экспорт» и в

появившемся окне ввести название и тег уровня, после чего он будет загружен в базу данных.

Так же возможно экспортировать сразу несколько файлов, для этого необходимо открыть окно экспорта, представленное на рисунке 3.2, выбрав соответствующий пункт в меню, открывающемся при нажатии на кнопку «Экспорт». Нажав на кнопку «Добавить файлы», пользователь увидит стандартный файловый диалог, который откроется в папке по умолчанию для сохранения уровней. Пользователь может выбрать один или несколько файлов и добавить их в окно экспорта. В окне экспорта будет создана таблица, в которой каждая строка соответствует одному добавленному уровню. Пользователь может изменить название или тег уровня в этом же окне, прежде чем загрузить уровни в базу данных. Так же пользователь может удалить из списка уровень нажав на соответствующую кнопку.

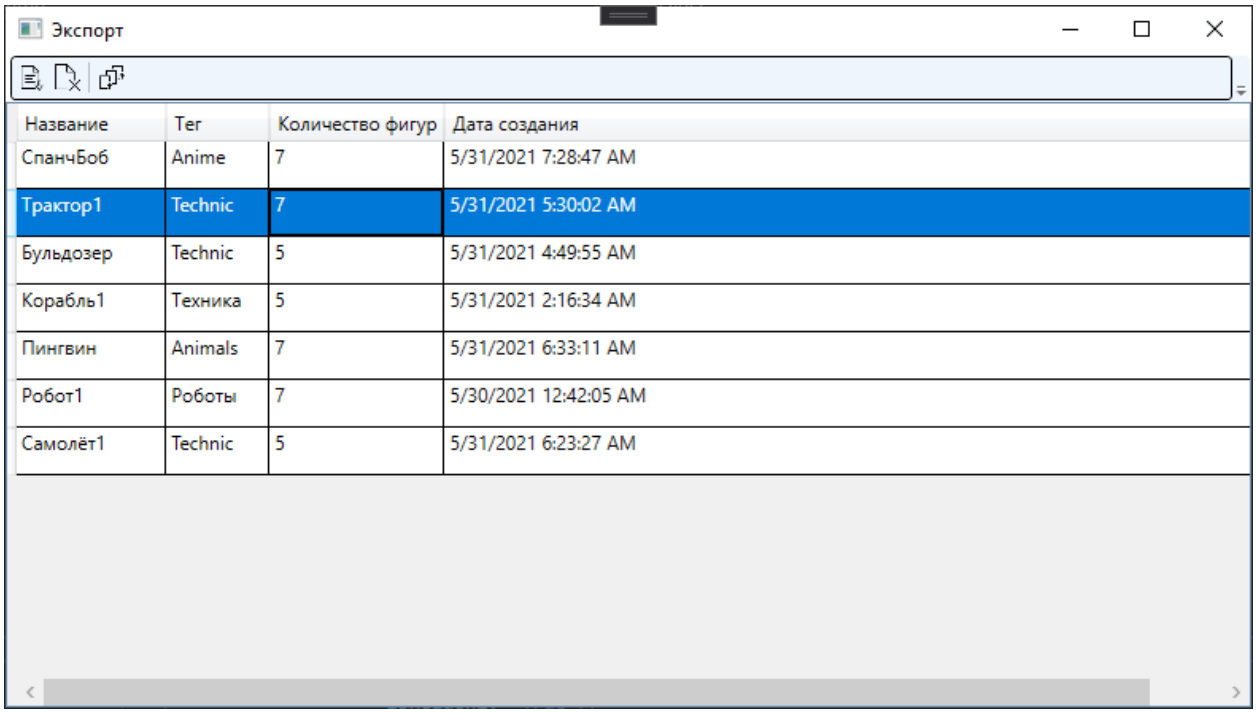

Рисунок 3.2 – Окно экспорта

Рисунок-уровень так же можно сохранить в формате png, чтобы перед добавлением в игру можно было согласовать его визуальный вид.

3.1.6 Перестроение фигур

Для того чтобы фигура могла изменяться при добавлении или удалении из неё примитивов для хранения примитивов используется ObservableCollection. Это динамическая коллекция объектов одного типа, которая генерирует событие CollectionChanged при добавлении или удалении из неё элемента. При возникновении такого события данные изображения, которым представлена фигура автоматически обновляются.

3.1.7 Зависимости объектов

В процессе создания рисунка-уровня формируется 2 вида зависимостей. Во-первых, каждая фигура содержит один или более примитивов, которые располагаются внутри этой фигуры определённым образом. Во-вторых, любые две фигуры образуют связь типа «главный-зависимый» при которой главная фигура хранит идентификатор зависимой фигуры, а зависимая идентификатор главной и разницу между своей начальной точкой и начальной точки главной фигуры. Это значение вычисляется при связывании фигур и не меняется при перестановке фигур в процессе создания рисунка-уровня.

3.2 Тестирование

Тестирование программы проводилось вручную. Например, для тестирования функции, перестраивающей примитивы внутри фигуры, после её перемещения были выполнены следующие действия:

- создана фигура, содержащая по одному экземпляру каждого типа примитива;
- фигура перемещена на 100 и 200 координат по осям X и Y соответственно;
- проверено, что координаты прямоугольника, ограничивающего каждый из примитивов, изменились на 100 и 200 по осям X и Y соответственно.
- проверено, что каждый из примитивов так же изменил свои координаты на 100 и 200 по осям X и Y соответственно.

Подобное тестирование было проведено для всех прецедентов, описанных в главе 1, а также для функций над фигурами, которые появились в процессе реализации приложения, например: на передний/задний план, отсоединить примитив, пересчитать точки привязки.

Было создано более 20 уровней различной сложности, все они были добавлены в игру. Каждый уровень был отрисован так, как ожидалось, сборка рисунка из частей осуществляется в соответствии с тем, как фигуры были связаны при создании рисунка-уровня в редакторе.

На рисунке 3.3 представлен скриншот из приложения «Картинки из геометрических фигур», в котором открыт один из созданных в редакторе уровней.

Рисунок 3.3 – Рисунок-уровень открытый в игре

3.3 Инструкция по установке

Для того чтобы пользователь мог без проблем получить и использовать приложение был создан установочный файл, который можно загрузить с облачного хранилища Google Диск [16]. Скачав файл, пользователю нужно будет запустить его и следовать инструкциям установщика. После установки приложение будет доступно для использования. После окончания установки, на рабочем столе появится ярлык, с помочью которого можно запустить приложение.

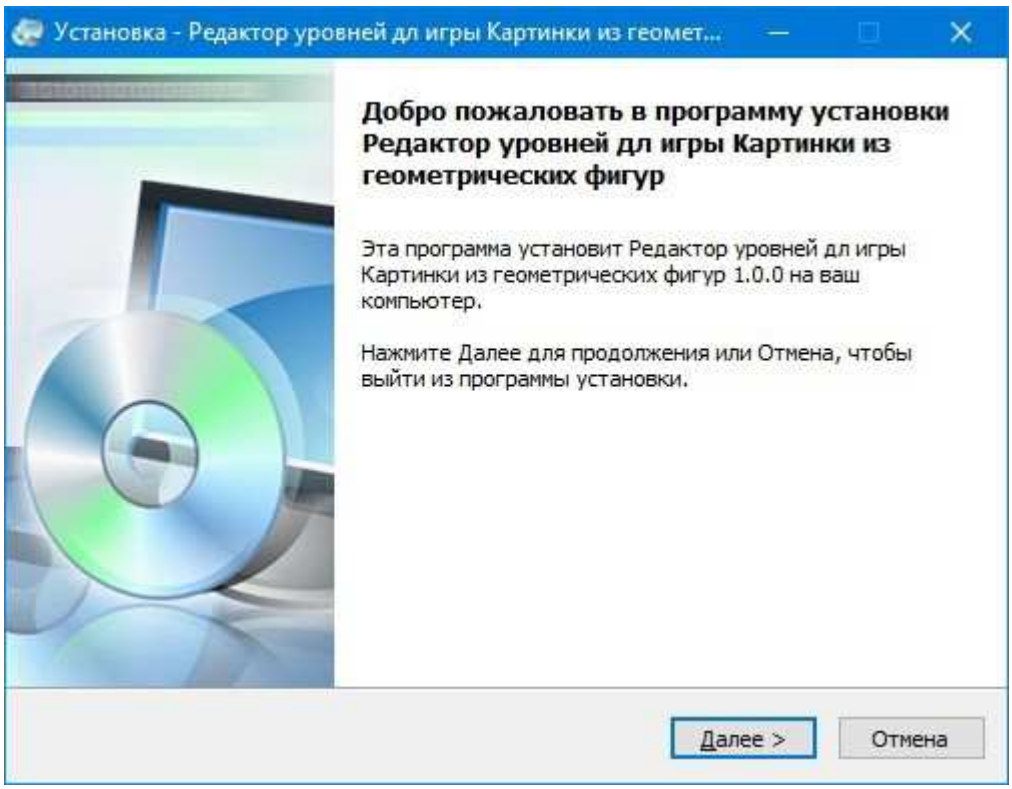

Рисунок 3.4 – Окно установщика

3.4 Инструкция разработчика

Для полноценной работы с WPF может быть использована как Visual Studio, так и Expression Blend [17]. Так как Expression Blend ориентирована на дизайн, рассмотрим вариант с Visual Studio.

Для того чтобы собрать и скомпилировать проект необходимо выполнить следующие действия:

- скачать Visual Studio 2019 [18] версии не ниже 16.9;
- в окне выбора компонентов, представленном на рисунке 3.5, выбрать пункт «Разработка классических приложений .NET»;
- запустить Visual Studio и в открывшемся окне, представленном на рисунке 3.6, выбрать «Клонирование репозитория»;
- ввести url-адрес git-репозитория [19] в соответствующее поле и нажать кнопку «Клонировать».

После этого проект клонируется на компьютер и будет доступен для редактирования и запуска в Visual Studio.

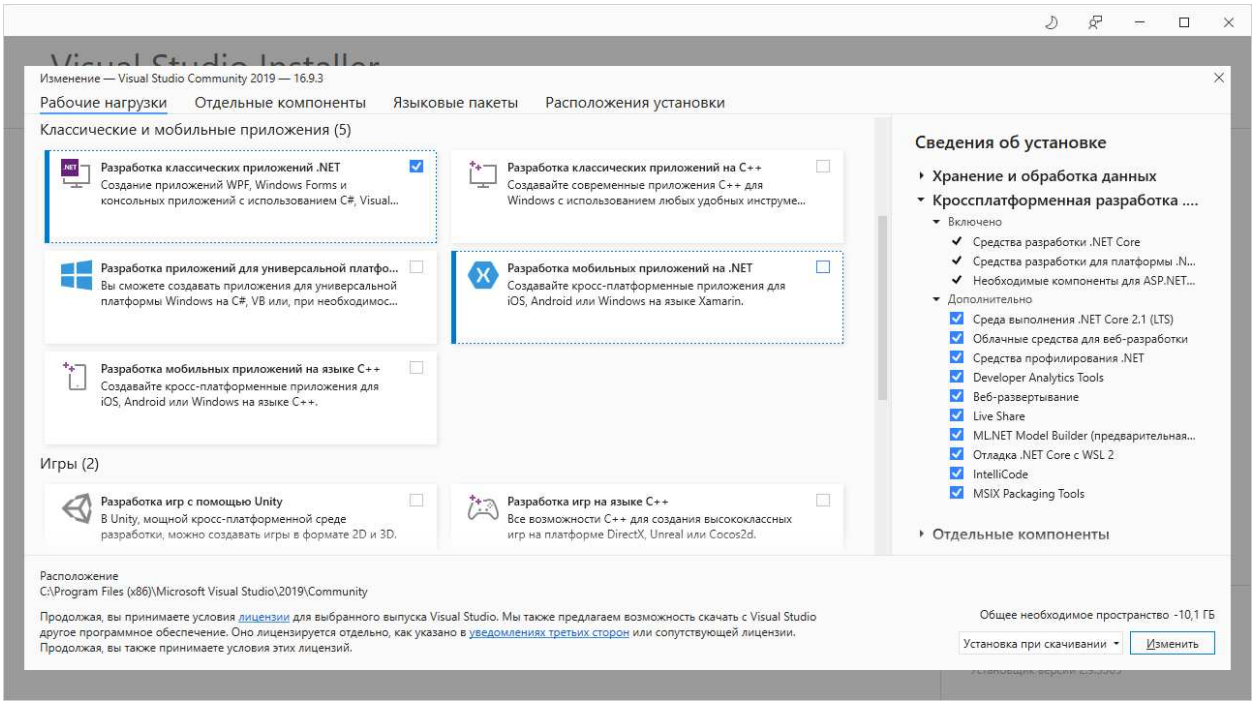

Рисунок 3.5 – Окно выбора компонентов

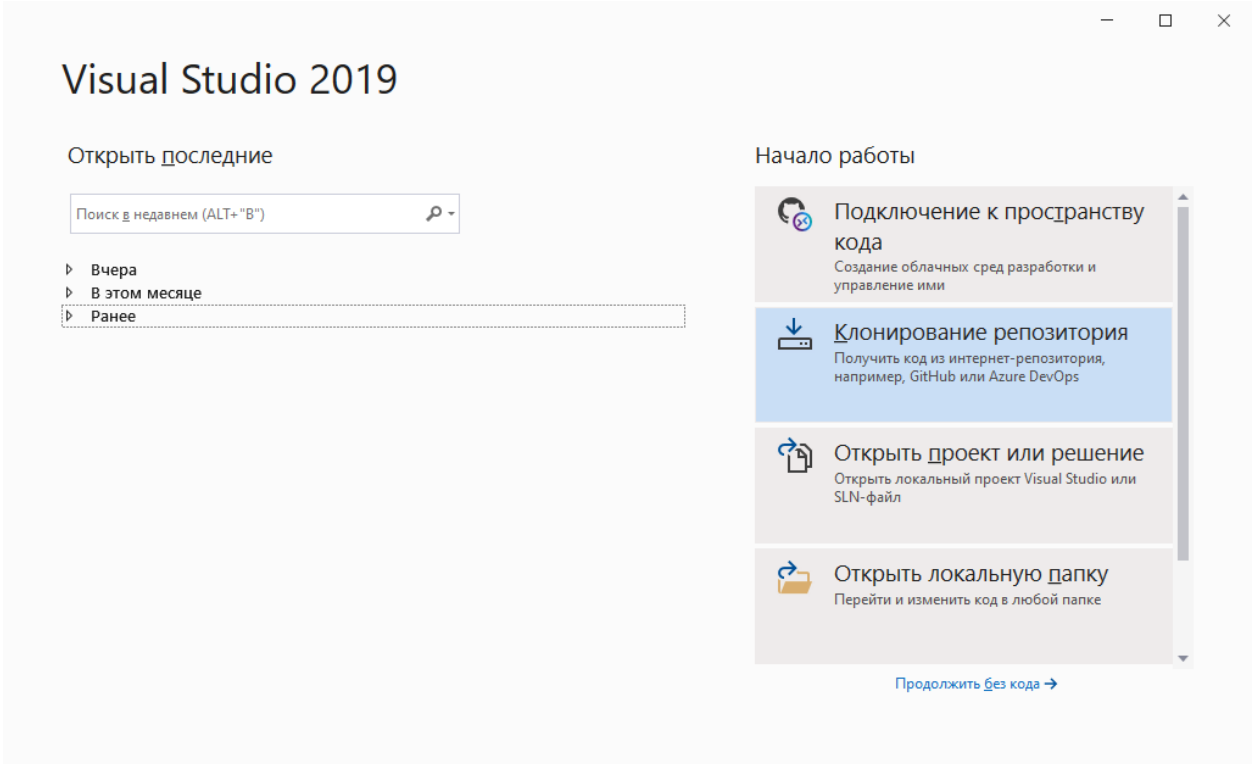

Рисунок 3.6 – Окно клонирования репозитория

3.5 Выводы по главе

- Разработан понятный, удобный и информативный интерфейс;
- реализовано взаимодействие с удалённой базой данных для хранения уровней, использующихся в игре;

- приложение протестировано;
- разработана инструкция для установки и запуска приложения;
- разработана инструкция разработчика, содержащая информацию для установки среды разработки и открытия проекта для редактирования и запуска.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы реализовано приложение «Редактор уровней для игры Картинки из геометрических фигур», которое позволяет создавать рисунки уровни из различных примитивов (прямоугольник, овал, линия, треугольник), так же реализована возможность рисования многоугольников и ломанных, примитивы объединяются в фигуры, которые можно связывать между собой, перемещать на холсте.

Для разработчика написана инструкция по установке рабочей среды и получения проекта для редактирования и запуска. Корректная работа приложения проверена путём ручного тестирования.

Приложение корректно функционирует на операционной системе Windows версии 7 и выше.

Проект доступен для скачивания из git-репозитория по ссылке [19].

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дидактическая игра "Собери картинку из геометрических фигур" // maam.ru URL: https://www.maam.ru/detskijsad/didakticheskaja-igra-soberi-kartinku-iz-geometricheskih-figur-991796.html (дата обращения: 04.02.2021).

2. Дидактическая игра «Сложи картинку из фигур» для детей 6-7 лет // sadikipermi.ru URL: https://sadikipermi.ru/files/igry/Картинки-к-игре-Сложи-картинку-из-фигур.pdf (дата обращения: 04.02.2021).

3. Рисунки из геометрических фигур - Задания в картинках и раскраски // bibusha.ru URL: https://bibusha.ru/risunki-iz-geometricheskikh-figur-zadaniya-raskraski (дата обращения: 04.02.2021).

4. Рисунок из геометрических фигур для детей: 1 класс, 2 класс (круг, овал, квадрат, треугольник и многоугольник) // montessoriself.ru URL: https://montessoriself.ru/risunok-iz-geometricheskih-figur-dlya-detej-1-klass-2-klass-krug-oval-kvadrat-treugolnik-i-mnogougolnik/ (дата обращения: 04.02.2021).

5. Танграм King // play.google.com URL: https://play.google.com/store/apps/details?id=com.mobirix.tangram (дата обращения: 06.02.2021).

6. Танграм Мастер // play.google.com URL: https://play.google.com/store/apps/details?id=com.littlebeargames.tangram (дата обращения: 06.02.2021).

7. Poly Shape - Tangram Puzzle Game // play.google.com URL: https://play.google.com/store/apps/details?id=com.easygame.simplepuzzle&hl=ru&gl=US (дата обращения: 06.02.2021).

8. Google Play // play.google.com URL: https://play.google.com/store?hl=ru&gl=US (дата обращения: 06.02.2021).

9. PlantUML // https://plantuml.com URL: https://plantuml.com/ru/ (дата обращения: 08.02.2021).

10. Процесс ICONIX. Диаграммы пригодности // pro-prof.com URL: https://pro-prof.com/archives/2723 (дата обращения: 10.02.2021).

11. Основы UML. Диаграммы последовательности // pro-prof.com URL: https://pro-prof.com/archives/2769 (дата обращения: 20.03.2021).

12. UML-диаграммы классов // prog-cpp.ru URL: https://prog-cpp.ru/uml-classes/ (дата обращения: 23.03.2021).

13. Общие сведения о WPF // docs.microsoft.com URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-5.0 (дата обращения: 20.05.2021).

14. Download .NET5 // dotnet.microsoft.com URL: https://dotnet.microsoft.com/download/dotnet/5.0 (дата обращения: 20.05.2021).

15. Общие сведения о привязке данных (WPF .NET) // docs.microsoft.com URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/data/?view=netdesktop-5.0 (дата обращения: 22.05.2021).

16. Game Level Redactor // Google Диск URL: https://drive.google.com/drive/folders/1s_TEdUN3U65Dyr5fr69vwpelr_-oXB3G?usp=sharing (дата обращения: 22.05.2021).

17. Microsoft® Expression® Studio Quick Start // www.microsoft.com URL: https://www.microsoft.com/en-us/download/details.aspx?id=4599 (дата обращения: 20.05.2021).

18. Visual Studio 2019 для Windows и Mac // visualstudio.microsoft.com URL: https://visualstudio.microsoft.com/ru/downloads/ (дата обращения: 20.05.2021).

18. Game Level Redactor // GitHub URL: https://github.com/mzsts/GameLevelRedactor (дата обращения: 22.05.2021).

Федеральное государственное автономное
Образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Вычислительная техника

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О.В. Непомнящий
подпись инициалы, фамилия
«_07_» _06_ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Редактор уровней для игры "Картинки из геометрических фигур"

Руководитель Васильев 08.06.2021 ст. преподаватель В.С. Васильев
подпись, дата

Выпускник Баландин 07.06.2021 М.А. Баландин
подпись, дата

Нормконтролер Васильев 08.06.2021 ст. преподаватель В.С. Васильев
подпись, дата

Красноярск 2021