**4 КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

В разделе 3 описана разработка архитектуры программного средства, включающая общий вид его структуры и порядок функционирования. Таким образом, определена основа, достаточная для реализации программы в виде исполняемого кода. Обычным средством для реализации является платформа разработки, позволяющая программистам работать с удобным набором команд вместо бинарного машинного кода.

**4.1** Выбор средств разработки

В подразделе 3.1 был определен общий вид программы - оконное приложение, а также характер выполняемых работ – обработка графической информации. Из этого следует ряд требований, определяющих выбор средств разработки:

- возможность создания графического интерфейса;

- поддержка объектно-ориентированного программирования;

- высокая скорость выполнения программного кода;

- наличие библиотек для работы с изображениями.

Первоочередной задачей является выбор языка программирования. Среди большого множества современных языков существует разделение на две группы - компилируемые и интерпретируемые, различающиеся методом исполнения кода. Для разрабатываемого ПС выбрана группа компилируемых языков, так как производительность программы находится в числе важнейших требований. Также язык должен быть объектно-ориентированным, что приводит к ограниченной группе популярных языков: C++, C#, Java, Visual Basic. Java использует особую среду JVM, позволяющую программам выполняться на машинах разных конфигураций, однако требует дополнительных вычислительных затрат, что делает этот язык нежелательным. C++, C# и Visual Basic объединены тем, что могут быть реализованы на одной платформе – Microsoft .NET. Она характеризуется высокой степенью интеграции различных технологий с помощью единого байт кода CIL (Common Intermediate Language) и библиотек DLL (Dynamic Link Library). Так, например, библиотека, написанная на языке Visual C++, может быть использована при написании программ на C# и Visual Basic. Это обеспечивает большую общую базу компонентов для повторного использования. Из трех языков C# наиболее прост в использовании и обеспечивает быструю реализацию объектно-ориентированных архитектур. По производительности он несколько уступает C++, однако этот недостаток может быть преодолен путем использования высокопроизводительных библиотек.

Еще одной характерной особенностью платформы .NET является привязанность к единой интегрированной среде разработки – Microsoft Visual Studio. Помимо редактора исходного кода, она предоставляет широкие возможности для построения схем, анализа кода, управления версиями, визуального программирования и многих других современных методик.

В стандартном пакете библиотек и фреймворков платформы .NET существует один, приспособленный для создания оконных приложений. Windows Forms представляет собой событийно-ориентированный фреймворк. Большая часть времени своей работы приложение ждет действий пользователя, предоставляя для этого графический интерфейс – «форму». В результате какого-либо действия – «события» в терминах фреймворка – исполняется ответственный за его обработку код. Windows Forms предлагает эффективные средства визуального программирования, позволяющие реализовывать графический оконный интерфейс за короткое время с написанием минимального количества кода (для обработки событий).

Платформа .NET имеет ряд встроенных библиотек для работы с графическими данными, однако они ориентированы на удобство и универсальность использования, и поэтому являются недостаточно производительными для обработки больших объемов данных. Для этих целей существует популярная платформонезависимая библиотека OpenCV (Open source Computer Vision). Она состоит из наборов высокопроизводительных функций для обработки изображений в реальном времени. Библиотека написана на языке C++, однако посредством оберточных интерфейсов доступна для языков C#, Perl, Ch, Ruby, Java и Python. Подобной оберткой в случае C# является Emgu CV. Так, через высокоуровневый интерфейс осуществляется доступ к эффективным функциям для обработки графических и статистических (что важно для разрабатываемого ПС) данных.

**4.2** Описание компонента синтеза панорамы

На аналитической модели приложения, представленной ранее на рисунке 3.1, видны основные элементы архитектуры и их зависимости друг от друга. Элемент «Приложение» - центр всех зависимостей, наиболее специфичный компонент программного средства. Исходящие из него зависимости имеют звездообразную топологию и почти не пересекаются. Это отражает уровни изоляции разных компонентов программы. При эффективном управлении зависимостями в архитектуре ее составные компоненты могут быть реализованы в виде модулей и библиотек, взаимозаменяемых и пригодных для повторного использования. В данной архитектуре можно четко выделить несколько изолированных групп компонентов:

- компонент, отвечающий за синтез панорамы из набора изображений;

- компонент, управляющий файлами;

- окно приложения, обеспечивающее основной графический интерфейс;

- графический редактор;

- компонент, управляющий настройками программы.

Среди перечисленных наиболее сложным и значимым для программного средства является компонент синтеза панорамы, так как он реализует основную функцию программы. Ввиду комплексного характера процесса построения панорамы и большого количества используемых структур данных и алгоритмов их обработки, важными условиями получения эффективного результата являются гибкость и простота реализации. Поэтому компонент разрабатывается в наиболее абстрактном виде, со строгими связями между его составляющими и возможностью для расширения. В рамках объектно-ориентированного программирования этой цели служат интерфейсы. В сочетании с приемом проектирования «Фабрика», они позволяют разделять разрабатываемые модули на две составляющие: связи между абстрактными сущностями и независимые друг от друга реализации этих сущностей.

Основная часть компонент синтеза панорамы представлена библиотекой Panoramas, хранящей связный набор интерфейсов. Дополнительно присутствует ряд классов, реализующих простейшие структуры данных и определяющих порядок генерации результата. Доступ к ним по интерфейсам реализует «Фабрика» библиотеки. Все интерфейсы класса и их зависимости представлены на рисунке 4.1.

Далее следуют описания каждого модуля библиотеки.

**4.2.1** Класс Stitcher

Класс Stitcher выполняет роль «фасада» библиотеки, принимая вызовы от приложения. Его интерфейс IStitcher устроен таким образом, чтобы приложение не знало о способе выполнения синтеза панорамы, т.е. не имело доступа к другим внутренним класса и интерфейсам библиотеки. Исключение представляет лишь интерфейс IRelationControl (см. подраздел 4.2.Х), являющийся средством влияния приложения на процесс синтеза. В иных случаях принимаемые и отдаваемые объектом класса Stitcher значения должны представлять стандартные или общедоступные структуры данных. Интерфейс IStitcher определяет два метода:



Рисунок 4.1 – Диаграмма классов библиотеки Panoramas

- Метод MatchBetween. Принимает имена изображений, участвующие в построении панорамы (они известны вызывающей метод программе, так передаются ею генератору панорамы в качестве исходных данных). Метод возвращает объект интерфейса IRelationControl, отображающий информацию о связи между изображениями и позволяющий редактировать ее в качестве настройки генерации. Вспомогательный метод, необязателен для нормального функционирования программы.

- Метод StitchAll. Не принимает параметров, возвращает представление панорамы в виде стандартного изображения (экземпляр класса Bitmap). Таким образом, заключает в себе главную функцию библиотеки.

Интерфейс реализован классом Stitcher, доступным через стандартную фабрику библиотеки Panoramas. Его атрибуты – основные компоненты библиотеки:

- объекты интерфейсов IAnalizer, IBuilder и IPresenter;

- представления панорамы в виде объектов IPanoramaImages, IPanoramaRelations и IPanoramaTransformations;

- объект фабрики IFactory.

Конструктор класса принимает экземпляр фабрики и массив изображений. Последние принадлежат нестандартному интерфейсу IImage, за реализацию которого отвечает фабрика (экземпляр Stitcher сам конструируется фабрикой, а не внешним приложением). По набору атрибутов класса прослеживается череда преобразований, через которые проходят данные при выполнении процесса склейки панорамы: анализ, сборка и представление результата. Через интерфейс класса доступны данные, являющиеся результатом анализа входных изображений, следовательно, нет причин выполнять анализ в отдельном от конструктора методе. Так, объекту класса Stitcher в дальнейшем всегда доступен экземпляр IPanoramaRelations, представляющий результаты анализа. При вызове MatchBetween осуществляется обращение к этому экземпляру. В результате обращения данные могут быть изменены, что следует учитывать при выполнении дальнейших этапов процесса синтеза панорамы.

Вызов метода StitchAll продолжает процесс синтеза, используя результаты анализа для сборки панорамы и ее представления в виде Bitmap. Переиспользовать результаты синтеза нельзя, так как данные могут быть изменены приложением через интерфейс IRelationControl. Если судить по фактическому набору методов класса Stitcher, обязательными атрибутами для него являются лишь объекты IBuilder, IPresenter и IPanoramaRelations. Тем не менее, наличие полного набора атрибутов наглядно отражает логику работы класса, а также позволяет менять ее с помощью расширения класса.

**4.2.2** Класс Segment

Segment - второй фундаментальный класс библиотеки, представляющий сегмент панорамы. При инициализации библиотеки любой сегмент представляет собой стандартное изображение класса Bitmap и его имя (идентификатор). С течением процесса генерации определяется место каждого сегмента на панораме и затем используется при сборке панорамы в виде Bitmap. То есть, начальные данные дополняются новыми, доступными на поздних этапах процесса. Поэтому класс Segment имеет два интерфейса: IImage и IImageTransformed. Первый определяет доступ к строковому имени Name и к изображению Bitmap. Второй расширяет IImage объектом интерфейса ITransformation, определяющим преобразование над изображением для его помещения на панораму. Реализация ограниченного и расширенного интерфейсов одним классом в данном случае оправдана малым объемом его содержания и высокой степенью изолированности (о существовании класса Stitcher знает лишь реализация DefaultFactory фабрики IFactory, все остальные модули используют его через интерфейсы).

**4.2.3** Класс PanoramaImages

Класс отвечает за хранение исходных данных процедуры построения панорамы – простых изображений. Как и Segment, класс доступен из фабрики DefaultFactory через его интерфейс IPanoramaImages. Данный интерфейс описывает лишь одно свойство – массив объектов интерфейса IImage. Объект класса PanoramaImages осуществляет инициализацию и хранение этого массива. Класс предельно прост, его роль – контроль доступа к коллекции исходных изображений. В контексте процесса синтеза его можно считать первой формой панорамы. В дальнейшем эта форма будет расширена.

**4.2.4** Класс PanoramaRelations

PanoramaRelations отвечает за хранение и доступ к результатам анализа исходных изображений. Его интерфейс IPanoramaRelations обусловлен вариантами дальнейшего использования этих результатов при построении панорамы. Он расширяет интерфейс IPanoramaImages и включает в себя:

- Список объектов Relations интерфейса IImagesRelation. Это вспомогательный элемент интерфейса, обеспечивающий его легкую расширяемость.

- Метод Core, возвращающий одно из изображений IImage, играющее роль «ядра» панорамы по критерию схожести сегментов между собой.

- Две перегрузки метода MatchBetween. Одна из них принимает в качестве параметров имена двух изображений и возвращает объект интерфейса IRelationControl, описывающий связь между ними. Такой вариант метода используется элементов Stitcher для предоставления дополнительного интерфейса пользователю библиотеки. Второй вариант метода принимает два объекта изображений IImage и возвращает экземпляр IImagesRelation. Функция метода та же, однако предоставляемый им объект предназначен для использования внутри библиотеки.

- Метод NeighboursOf. Принимает изображение IImage и массив изображений, из которых выбираются наиболее схожие с данным. При отсутствии массива выбор осуществляется из всего множества сегментов панорамы.

- Метод ClosestBetween. Имеет сложный набор параметров: два набора изображений и две ссылки IImage для возврата результатов. Предназначен для нахождения пары наиболее схожих изображений из двух предоставленных множества.

Класс PanoramaRelations представляет результаты операции анализа над исходной коллекцией изображений, и поэтому является наследником класса PanoramaImages, расширяя его дополнительными данными (массив объектов IImagesRelation) и операциями над ними. Также наследование обеспечивает беспрепятственный доступ к набору исходных данных. В контексте процесса синтеза, PanoramaRelations – второе представление панорамы, дополненное связями между ее сегментами.

Выбор «ядра» панорамы в методе Core осуществляется путем поиска изображения с наименьшим суммарным расстоянием от него до других. Подобные операции легко реализуются на языке C# с помощью стандартной библиотеки LINQ. Расстояние для каждого изображение определяется с помощью закрытого метода distancesFor. Схожим образом реализованы методы MatchBetween: последовательным перебором находится экземпляр связи, соответствующий переданным параметрам.

Метод ClosestBetween с помощью операций LINQ находит все связи, к которым принадлежат изображения первого из переданных ему наборов, затем выбирает из них связи со вторым набором и сортирует в порядке увеличения схожести. Первый объект в этой коллекции – искомая связь между двумя наборами изображений.

Метод NeighboursOf выбирает те исходные изображения, самым схожим для которых является данное.

**4.2.5** Класс PanoramaTransformations

PanoramaTransformations отвечает за хранение и доступ к результатам построения панорамы. Его интерфейс IPanoramaTransformations включает лишь массив объектов Segments интерфейса IImageTransformed (см. подраздел 4.2.2), то есть сегменты панорамы с определенными расположениями на ней. Объект класса инициализирует свой массив и предоставляет доступ к сегментам. Он расширяет PanoramaRelations, представляя собой последнюю итерацию данных панорамы, готовую к представлению.

**4.2.6** Интерфейс IImagesRelation

Интерфейс представляет связь между двумя изображениями, предназначенную для использования при построении панорамы. Элементы интерфейса:

- Экземпляры BaseImage и QueryImage интерфейса IImage. Представляют два изображения, отношение между которыми описывает объект интерфейса.

- Экземпляр ReversePair того же интерфейса IImagersRelation. Ссылается на объект, описывающий обратную связь изображений. Такое разделение по направлению связи необходимо ввиду неравнозначности их отношений друг к другу.

- Метод Includes. Принимает объект IImage и возвращает логическое значение, говорящее, состоит ли изображение в данной связи.

- Метод PairOf. Принимает объект IImage и возвращает связанный с ним данным соотношением.

- Метод GenerateTransformation. Возвращает объект интерфейса ITransformation, как средство реализации связи двух изображений.

**4.2.7** Интерфейс ITransformation

Описывает преобразование какого-либо изображения. В контексте процедуры синтеза панорамы, является способом размещения отдельного сегмента на ней. Методы интерфейса:

- Метод Reset. Служит для установки преобразования в свое изначальное состояние. Изначально объект интерфейса должен описывать такое преобразование, которое не изменяет положение изображения.

- Метод Move. Принимает две целочисленные величины, определяющие смещение изображения в пределах плоскости двухмерного пространства.

- Метод Multiply. Принимает другое преобразование и возвращает результат его объединения с текущим. Метод необходим для работы с последовательностями преобразований.

- Метод Distort. Принимает другое преобразование и объединяет текущее с ним.

- Метод TransformOn. Применяет преобразование к данному изображению стандартного класса Bitmap и размещает его на другом. Ожидаемый результат – второй параметр (холст) с нанесенным на него изображением.

**4.2.8** Интерфейс IAnalyzer

IAnalyzer отвечает за процедуру анализа исходных изображений и представление его результатов в виде отношений. Включает метод Analyze, который принимает набор изображений (IPanoramaImages) и возвращает коллекцию результатов анализа (IPanoramaRelations). Модуль не имеет стандартной реализации в библиотеке Panoramas и предназначен для использования при расширении библиотеки. Причина этого – отсутствие стандартного алгоритма или метода анализа.

**4.2.9** Интерфейс IBuilder

IBuilder отвечает за процедуру построения панорамы, выражаемую в определении положения каждого из связанных между собой изображений. Включает метод Build, принимающий набор изображений и их связей (IPanoramaRelations) и возвращающий набор позиционированных изображений (IPanoramaTransformations).

**4.2.10** Интерфейс IPresenter

IPresenter – последний обработчик данных панорамы, представляющий ее в доступном для приложения варианте – в виде экземпляра класса Bitmap. Аналогично IAnalyzer и IBuilder, не имеет стандартной реализации ввиду отсутствия универсального способа представления сегментов панорамы на одном изображении. Интерфейс состоит из одного метода – Present, принимающего набор позиционированных изображений (IPanoramaTransformations) и возвращающий объект Bitmap.

**4.2.11** Интерфейс IRelationControl

Представляет дополнение к «фасаду» библиотеки – IStitcher - для управления настройками генерации панорамы и их наглядного представления. В качестве объекта настроек выступает связь между двумя исходными изображениями. Состав интерфейса:

- Метод ToImage. Возвращает представление связи между изображениями в виде экземпляра стандартного класса Bitmap.

- Метод Similarity. Возвращает целочисленное значение используемой метрики схожести двух изображений. Значение должно отображать результаты изменения настроек приложением.

- Целочисленное значение LimitPercent. Предоставляет доступ к параметру, описывающему в процентах некоторую характеристику связи (в конкретном случае – полноту используемого набора особых точек).

- Флаг Active. Простейший элемент интерфейса, позволяет исключать связь из использования при построении панорамы.

Реализация интерфейса относится к конкретным методам анализа и сборки панорамы, поэтому в библиотеке отсутствует.

**4.2.12** Класс DefaultFactory

DefaultFactory – фабрика, используемая для расширения библиотеки Panoramas и реализации ее компонентов. Фабрика доступна внутренним модулям библиотеки через интерфейс IFactory, включающий методы:

- Метод Image. Возвращает объект интерфейса IImage, полученный на основе данного строкового имени и изображения Bitmap.

- Метод Transformation. Возвращает объект интерфейса ITransformation (в исходном состоянии).

- Метод ImageTransformed. Возвращает объект IImageTransformed, образованный данным объектом IImage и описывающим его положение объектом ITransformation.

- Метод PanoramaImages. Принимает массив исходных изображений и IImage возвращает использующий их объект интерфейса IPanoramaImages.

- Метод PanoramaRelations. Принимает контейнер изображений IPanoramaImages и список описывающих их связей IImagesRelation. Возвращает объект представления связных изображений IPanoramaRelations.

- Метод PanoramaTransformations. Принимает контейнер IPanoramaRelations и массив позиционированных изображений IImageTransformed. Возвращает контейнер позиционированных изображений IPanoramaTransformations.

- Метод Stitcher. Принимает массив изображений IImage и возвращает экземпляр IStitcher.

- Методы Analyzer, Builder и Presenter. Не принимают параметров, возвращают экземпляры интерфейсов IAnalyzer, IBuilder и IPresenter соответственно.

Класс DefaultFactory включен в состав библиотеки Panoramas, однако реализации некоторых элементов его интерфейса в ней не определены. В языке C# существует механизм частичной реализации интерфейсов – абстрактные классы. Подобные классы не имеют конструкторов и требуют своего расширения. Как интерфейсы, они способы определять методы для расширения (абстрактные), однако также могут самостоятельно реализовывать другие методы. Следовательно, целесообразно сделать DefaultFactory абстрактным классом с реализацией методов Image, ImageTransformed, PanoramaImages, PanoramaRelations, PanoramaTransformations и Stitcher, так как требуемые ими классы присутствуют в библиотеке (описаны выше). Реализации представляют собой инициализацию объектов этих классов с передачей им параметров. Остальные методы интерфейса – IAnalyzer, IBuilder, IPresenter и ITransformation – описаны классом как абстрактные, т.е. требующие реализации при наследовании.

Описанный интерфейс IFactory предназначен для использования внутренними модулями библиотеки и их расширениями. Предоставление его объекта приложению было бы нарушением принципа изолированности реализаций. Поэтому, определен еще один интерфейс фабрики – IPublicFactory. Он предназначен для внешнего использования, и содержит только один метод – Stitcher. Параметры метода – массив строковых имен и массив изображений Bitmap. Возвращаемое значение – объект интерфейса IStitcher. Таким образом, интерфейс IPublicFactory позволяет ограничить использование фабрики инициализацией главного интерфейса библиотеки. Реализация метода в классе DefaultFactory проверяет соответствие переданных массивов по их размерам, конструирует на их основе объекты IImage и использует метод Stitcher внутреннего интерфейса IFactory для инициализации IStitcher.