**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 компонентов и классов программы

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 используемых повторно. 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.

Далее они будут рассмотрены подробно.

**4.2.1** Компонент синтеза панорамы

Компонент реализован библиотекой Panoramas. Она отвечает за реализацию главной функции программы – синтез панорамы как цельного представления множества изображений. Следовательно, входные данные – изображения в общем представлении (пути к файлам, объекты стандартных классов), выходные – одно изображение, также в общепонятном представлении. Также необходим элемент управления, с помощью которого приложению (и пользователю) будет доступен просмотр и изменение настроек генерации результата.

Неоднородность содержания компонента и сложные связи между его модулями говорят о целесообразности применения приема проектирования, известного как «Фасад». Он заключается в создании класса, полностью обеспечивающего интерфейс между компонентом и приложением. В качестве фасада разработан класс Stitcher. Члены класса Stitcher:

- Конструктор. Имеет две перегрузки, принимающие либо список имен файлов, которые впоследствии будут загружены объектом класса, либо массив изображений в виде класса стандартного класса Bitmap, а также их имен (для удобства идентификации). Передача непосредственно изображений позволяет избежать потерь времени, требуемого на загрузку и обработку данных с жесткого диска. Оба конструктора принимают списки неограниченной длины, однако не меньшей 2. Генерация панорамы из одного изображения не имеет смысла, так как оно уже несет в себе всю доступную информацию. В алгоритме работы программы (подраздел 3.3.1) в процессе синтеза панорамы были определены два четких этапа – анализ сегментов и собственно генерация результата. Вынесение процедуры анализа в интерфейс класса Stitcher усложнило бы его, т.к. назначение и значение метода, его необходимость для последующего вызова StitchAll, являются неочевидными. Выполнение анализа в методе StitchAll противоречит алгоритму работы программы и концентрирует временные затраты в вызове одного метода. Поэтому решено выполнять анализ сразу при инициализации объекта Stitcher и получении им исходных изображений.

- Метод StitchAll выполняет генерацию панорамы. Метод не принимает аргументов, возвращает панораму в самом общем представлении, в виде объекта Bitmap.

- Метод MatchBetween обеспечивает доступ к результатам анализа исходных изображений. Он принимает два имени файла и возвращает экземпляр реализации специального открытого интерфейса IImagesMatch. При инициализации объект класса Stitcher получает имена файлов всех исходных изображений и, следовательно, может в дальнейшем идентифицировать их. MatchBetween осуществляет поиск полученной в результате анализа связи между указанными изображениями.

Интерфейс IImagesMatch определяет, какими параметрами генерации может управлять пользователь. Он должен обеспечивать минимальный необходимый поток информации, не нарушающий правил изоляции компонента Panoramas. Элементы интерфейса:

- Свойство CurrentPercent. Отвечает за количество используемых пар особых точек, в порядке убывания их схожести. Значения свойства могут быть в пределах от 0 до 100, независимо от реального количества точек. Таким образом интерфейс ограничивает ненужную выдачу информации внутреннего пользования. Начальное значение CurrentPercent будет отображать автоматически определенное программой количество схожих пар точек.

- Свойство Active. Флаг, отвечающий за факт использования связи сегментов при генерации панорамы. Пользователь может выключать его, если считает, что сегменты несовместимы друг с другом.

- Метод ToImage. Первоочередным способом отображения информации и связи между сегментами панорамы решено использовать графическое представление. Оно позволяет ограничить количество доступных параметров и одновременно представить их совокупно в наглядном виде. Ввиду специфики используемых методов анализа сегментов, предполагается, что изображение будет содержать информацию об особых точках и связях между ними.

Класс Segment - один из ключевых классов компонента Panoramas. Объекты класса Segment недоступны через интерфейс модуля. Определенные как элемент типа «Сущность» (подраздел 3.1), они служат для хранения и управления доступом к данным. Так как Segment является главным представлением исходного изображения, он содержит его оригинальное представление Bitmap, а также имя Filename. Помимо этого, каждый сегмент в результате генерации панорамы должен иметь определенную позицию на ней или, в терминах используемой методологии, трансформацию по отношению к глобальному изображению панорамы. Эта информация представлена свойством Transformation. Данное свойство – единственное в классе, которое может быть изменено (в результате перенастройки связей между сегментами). Этой цели служит метод ResetTransformation – безопасный способ вернуть сегмент в свое исходное положение.

Класс Transformation описывает отношение преобразования между двумя изображениями. Он реализует возможность операций над преобразованием и его применения к данному изображению. В качестве основы взят класс HomographyMatrix библиотеки Emgu CV, так что Transformation является его оберткой, изолирующей остальную часть модуля от подобной реализации. Члены класса:

- Конструктор. Не принимает аргументов, инициализирует матрицу преобразования начальными значениями.

- Метод Distort. Принимает другой объекта класса в качестве аргумента и изменяет текущий таким образом, что он будет представлять совокупное преобразование.

- Метод Transform. Принимает стандартное изображение (Bitmap) и возвращает его копию в преобразованном виде.

- Метод TransformOn. Преобразует данное изображение Bitmap и отображает его на данном холсте специального типа Emgu.CV.Image. Данный метод будет использоваться для быстрой генерации панорамы в виде изображения.

- Метод Move. Принимает значения прироста координат двухмерного пространства и корректирует преобразование соответствующим образом (линейное перемещение).

- Метод Clone. Корректно создает копию текущего объекта, с учетом специфики HomographyMatrix.

- Метод класса Generate. Определяет преобразование на основании данной пары сегментов. Использует высокоскоростной метод FindHomography библиотеки OpenCV. Возвращает экземпляр класса.

- Закрытый метод newHomography. Инициализирует новый экземпляр класса HomographyMatrix необходимыми значениями и возвращает его.

Так, определены базовые классы компонента Panoramas, определяющие его интерфейс и способы хранения информации. Далее требуется реализовать процедуры анализа сегментов и генерации панорамы. Они имеют четкое разделение в плане используемых процедур, поэтому будут представлены двумя отдельными модулями. Первоочередной процедурой является анализ, поэтому ее модуль – Matching – будет описан первым.

Класс SegmentsPair отвечает за получение, хранение и обработку информации об отношении двух сегментов панорамы. Он реализует интерфейс IImagesMatch, предоставляя часть своего функционала вне компонента Panoramas. Члены класса:

- Свойства BaseSegment и QuerySegment. Сегменты неравнозначны при определении отношения между ними, и используются соответствующим образом.

- Свойство Matches. Представляет массив пар особых точек между сегментами, упорядоченный по убыванию схожести. Это основные данные об отношении сегментов.

- Свойство Active. Флаг, указывающий, будет ли отношение между сегментами учтено при построении панорамы. Значения флага синхронизированы между текущим экземпляром и обратным к нему (установленным с помощью SetReversePair).

- Свойство LimitPercent. Определяет количество пар особых точек, определяющих отношение сегментов. Также синхронизировано.

- Конструктор. Принимает два сегмента и определяет отношение между ними, используя реализацию алгоритма KNN для соотнесения особых точек.

- Метод SetReversePair. Устанавливает ссылку на другой экземпляр класса, описывающий отношение между теми же сегментами, но в обратном порядке (т.е. основной сегмент в данном экземпляре является относительным в другом экземпляре). Ссылка используется при изменении отношения между сегментами.

- Метод ToImage. Возвращает изображение (Bitmap), хранящее информацию об отношении между сегментами. Делегирует представление своих данных объекту класса PairPresenter.

- Метод Distance. Возвращает значение метрики схожести двух сегментов. Метрика основана на метрике схожести особых точек и представляет совокупность ее значений для используемых пар точек.

- Метод GenerateTransformation. Один из основных методов класса, генерирует преобразование относительного сегмента в базовый. Использует статический метод Transformation.Generate.

- Метод Includes. Принимает экземпляр класса Segment, определяет, относится ли он к сравниваемой паре сегментов, возвращает логический признак.

- Метод PairOf. Также принимает Segment, возвращает его пару, если сегмент состоит в данном отношении, либо пустую ссылку (null).

- Закрытый метод setOptimalLimit. Устанавливает количество используемых пар особых точек на минимум, чтобы получить лишь наиболее точные данные об отношении сегментов.

- Закрытый метод setCountLimit. Принимает целочисленный аргумент как количество пар особых точек.

- Закрытый метод resetDistance. Обновляет значение метрики схожести двух сегментов на основании схожести пар особых точек.

Класс MatchingController отвечает за полноту анализа всех сегментов из данного набора, а также обрабатывает множество результатов анализа. Его члены:

- Конструктор. Принимает массив сегментов и инициирует процедуру анализа, так как весь интерфейс класса требует его результатов. Процедура реализована методом generateMatches, описанном ниже.

- Статический метод generateMatches. Принимает массив сегментов, возвращает массив всех отношений между ними. Для определения отношений метод осуществляет полный перебор сегментов, связывая при этом разнонаправленные отношения между собой.

- Метод MatchBetween. Имеет две перегрузки, принимающие либо пару сегментов, либо пару имен файлов. В любом варианте осуществляет поиск отношения между сегментами и возвращает результат.

- Метод CoreSegment. Возвращает сегмент, характеризуемый наибольшей близостью к нему других сегментов. Метод используется для определения главного сегмента панорамы.

- Метод ClosestTo. Принимает сегмент и возвращает другой, наиболее близкий (схожий) с данным. Имеет перегрузку с указанием массива сегментов. В таком варианте метод ищет ближайший сегмент среди данного массива.

- Метод NeighboursOf. Принимает сегмент и возвращает массив других, ближайшим к которым является данный. Ключевой метод при генерации дерева панорамы.

Последний класс модуля Matching – PairPresenter. Он отвечает за наглядное представление отношения двух сегментов в виде стандартного изображения Bitmap. Отображение реализовано следующим образом: изображения двух сегментов совмещаются на одном, на них указываются особые точки и соединяются между собой при наличии соответствия. Помимо конструктора, принимающего экземпляр SegmentsPair, имеет всего один общедоступный метод – Render, выполняющий главную функцию. Закрытые методы класса:

- Метод generateCommonTemplate. Возвращает стандартный Bitmap для размещения на нем обоих сегментов. Его размер определяется суммарным размером сегментов.

- Метод drawFeature. Принимает набор аргументов для отображения особой точки на общем изображении в виде окружности. Радиус окружности определяется размером особой точки.

Таким образом, модуль Matching предоставляет отношения между данными сегментами. Эти отношения в дальнейшем используются для генерации панорамы модулем Tree. Как следует из названия, модуль содержит специфическую реализацию метода генерации при помощи структуры данных типа «дерево». Далее следует описание его классов.

Класс TreeNode представляет узел дерева, его основу. Он отвечает за хранение данных и связей с другими узлами. Члены класса:

- Свойство Segment. Определяет хранимые узлом данные – сегмент панорамы.

- Свойства Parent и Children. Описывают связи узла с другими. Согласно логике построения дерева, у любого узла может быть только один «родительский» узел (выше по иерархии) и множество «дочерних» (ниже).

- Метод AddChild. Принимает экземпляр класса Segment. Создает новый узел для его хранения, добавляет его в набор своих дочерних и возвращает.

- Метод FindNode. Также принимает сегмент. Рекурсивно ищет узел, хранящий данный сегмент, и возвращает его (или null, в случае отсутствия такого).

- Метод CollectSegments. Рекурсивно собирает сегмент текущего и всех дочерних узлов в массив и возвращает его.

- Метод Transformation. Вычисляет положение сегмента на панораме путем сложения всех преобразований между ним и корнем дерева.

Класс TreeBuilder отвечает за построение дерева. Обладает минимальным интерфейсом – конструктор, принимающий MatchingController, и общедоступный метод Generate, возвращающий корень дерева. Алгоритм построения описан в подразделе 3.3.2. Экземпляр MatchingController используется для получения данных об отношениях между сегментами. Закрытые методы класса TreeBuilder:

- Метод addNodeToTree. Рекурсивная процедура, принимающая в качестве аргументов сегмент и узел дерева. Сегмент добавляется к дочерним элементам узла, становясь частью дерева. Далее определяются другие сегменты, ближайшие к данному, и для них процедура повторяется (с продвижением дальше по уровню иерархии).

- Метод resetSegments. Совершает проход по имеющемуся набору сегментов с установкой их преобразований (Transformation) в исходное состояние.

Класс TreePresenter отвечает за представление дерева в виде стандартного изображения – панорамы. Конструктор принимает корень дерева – экземпляр класса TreeNode. Открытый метод Render генерирует изображение Bitmap. Осуществляется это путем создания большого холста типа Emgu.CV.Image и рекурсивного отображения на нем всех сегментов при обходе дерева. Порядок обхода не критичен, однако в данной реализации используется прямой. Результат – холст большого размера с панорамой – переводится в стандартный Bitmap и обрезается по краям (так как был создан с запасом пространства). Закрытые методы класса TreePresenter:

- Метод renderNode. Принимает в качестве аргументов узел дерева, глобальную трансформацию и холст для рисования. Рекурсивно отображает узел и все его дочерние узлы, с учетом трансформации.

- Метод generateTemplate. Принимает изображение центрального сегмента панорамы и на основании его создает экземпляр Bitmap большого размера, который будет использоваться как холст для рисования панорамы.

**4.2.2** Компонент управления файлами

Компонент представлен библиотекой ImageFileManager. Он отвечает за загрузку, управление и отображение набора исходных изображений. Функции компонента мало зависят друг от друга, поэтому, в отличие от Panoramas, «фасад» здесь не нужен. Центральный класс – CollectionManager – является оболочкой коллекции файлов, а все остальные – вспомогательные средства интерфейса этой коллекции.

Класс ImageFile хранит соответствие между именем файла и его содержимым. Причина использования подобного класса – отсутствие имени файла в стандартном классе Bitmap. Класс имеет два свойства – FileName и Bitmap, и конструктор, задающий их значения.

Класс CollectionManager был упомянут выше как оболочка коллекции файлов. Его задачи – добавлять, удалять и представлять элементы коллекции. Также класс может управлять набором представителей – элементами управления, отображающих актуальное содержимое коллекции. Члены класса CollectionManager:

- Свойство Images. Список экземпляров класса ImageFile.

- Конструктор. Не принимает аргументов, инициализирует списки Images и presenters.

- Метод LoadMore. Принимает в качестве аргумента ссылку на процедуру. Метод вызывает диалоговое окно выбора файлов на жестком диске и, если файлы были выбраны, вызывает процедуру с передачей имен файлов. До вызова процедуры выбранные изображения добавляются в коллекцию. Вызывает обновление представителей коллекции.

- Метод ClearAll. Удаляет все элементы из коллекции. Вызывает обновление представителей коллекции.

- Метод Remove. Принимает массив имен файлов и удаляет из коллекции элементы с данными именами. Вызывает обновление представителей коллекции.

- Методы PresentAsViewList и PresentAsPairsList. Осуществляют отображение коллекции файлов на стандартных элементах формы. В дальнейшем при любых изменениях в коллекции данные элементы формы будут автоматически обновлены до актуального состояния. Методы возвращают объект интерфейса ISelectableControl.

Интерфейс IRefreshablePresenter. Определяет один метод – Refresh, принимающий актуальный набор изображений. Предназначен для обновления управляемых элементов формы (представителей коллекции).

Интерфейс ISelectableControl обеспечивает ограниченный единообразный доступ к элементам формы, с которыми связан CollectionManager. Он необходим по причине комплексного характера некоторых управляемых элементов. Члены интерфейса:

- Метод SelectedItems. Возвращает имена выбранных на элементе изображений.

- Метод AddSelectionChangeHandler. Принимает стандартный объект EventHandler и вызывает его при изменении набора выбранных изображений.

Класс Dialog реализует упрощенный доступ к стандартным классам SaveFileDialog и OpenFileDialog. Члены класса:

- Свойства SaveFileDialog и OpenFileDialog. Используемые экземпляры диалогов, доступные для настройки извне.

- Свойства MinimalFilesCount и MaximalFilesCount. Хранят минимальное (максимальное) количество файлов, которые могут быть выбраны с помощью диалогов.

- Конструктор. Инициализирует экземпляры диалогов.

- Методы OpenFile и OpenFiles. Вызывают появление диалога выбора файлов для последующей загрузки. Принимают ссылку на функцию, которой передают имя (имена) выбранных файлов.

- Метод SaveToFile. Вызывает диалог выбора файла для сохранения. Принимает ссылку на функцию, которой передает имя выбранного файла.

Библиотека ImageFilesManager также реализует два представителя коллекции изображений. Первый – класс ThumbnailsList, осуществляющий управление стандартным элементом формы ListView. Позволяет отображать файлы в виде блочного списка наглядных значков, что является оптимальным интерфейсом для управления загруженными файлами. Члены класса:

- Конструктор. Принимает экземпляр списка ListView, настраивает его и связывает с набором изображений ImageList.

- Метод RefreshWith. Реализация интерфейса IRefreshablePresenter, принимает набор изображений и последовательно добавляет их в список.

- Методы SelectedItems и AddSelectionChangeHandler. Реализация интерфейса IRefreshablePresenter. Делегируют управляемому списку получение массива выбранных элементов и установку обработчика.

- Метод Clear. Очистка списка.

- Метод fitImage. Закрытый метод, преобразующий стандартный Bitmap в удобный формат для добавления в ImageList. Необходим, так как по умолчанию ImageList искажает добавляемые в него изображения.

Второй представитель коллекции – класс PairsList. Он управляет двумя списками ListBox, таким образом, что в один момент времени на них выбраны разные пункты. Это необходимо для интерфейса управления генерацией панорамы, который отображает связь между двумя разными сегментами. Члены класса:

- Свойства MainItem и SubItem. Предоставляют доступ, соответственно, к главному и подчиненному списку.

- Конструктор. Принимает два экземпляра класса LixtBox. Инициализирует управляемые списки и добавляет к ним правила уникальности.

- Метод RefreshWith. Реализация интерфейса IRefreshablePresenter, принимает набор изображений и последовательно добавляет их имена в списки.

- Методы SelectedItems и AddSelectionChangeHandler. Реализация интерфейса IRefreshablePresenter. Делегируют управляемым спискам получение массива выбранных элементов и установку обработчика.

- Метод AddItems. Принимает массив строк, которые добавляются в списки.

- Метод Clean. Удаляет все хранимые элементы списков.

- Метод refreshMainList. Закрытый метод, заполняющий главный список элементами и устанавливающий текущий индекс на первый из них.

- Метод refreshSubList. Делает то же для подчиненного списка, однако не включает в него элемент, выбранный в главном.

- Методы selectedMainIndexChanged и selectedSubIndexChanged. Устанавливают правило уникальности: при совпадении элементов на обоих списках, на подчиненном выбирается другой элемент, а совпадающий – удаляется.