Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра Системного программирования

Курсовая работа

Обучающая программа

для медицинского тренажера «Максим»

Научный руководитель:

Дубчук Н. П.

Выполнил:

Студент 345 группы Титов А. Ю.

Санкт-Петербург

2010

Оглавление

Введение

В различных областях медицины создаются новейшие тренажеры для обучения и проверки знаний молодых специалистов, а также подтверждения квалификации медицинских работников.

Как правило, завод-изготовитель тренажёра снабжает его всеми необходимыми датчиками и сенсорами, а также простейшим программным обеспечением. Обычно данное ПО не позволяет удобно работать с тренажёром, а лишь даёт возможность отслеживать состояние его датчиков на компьютере, что не позволяет увидеть наглядной картины происходящего. Для улучшения качества обучения с использованием тренажёров возникает потребность в программном обеспечении, которое будет обладать интерактивностью, наглядно и в понятной каждому пользователю форме отражать происходящие в процессе выполнения действия. Помимо этого оно должно работать на большом количестве довольно старых компьютеров, так как многие учреждения, использующие в процессе обучения тренажёры, не обладают достаточно хорошей материальной базой для использования высокотребовательного ПО. В данной курсовой работе рассматривается решение задачи создания программного обеспечения для обучающего тренажёра сердечно-лёгочной реанимации «Максим-III-01»[1] (далее будем называть его просто «Максим»). В связи с перечисленными выше требованиями были поставлены следующие цели:

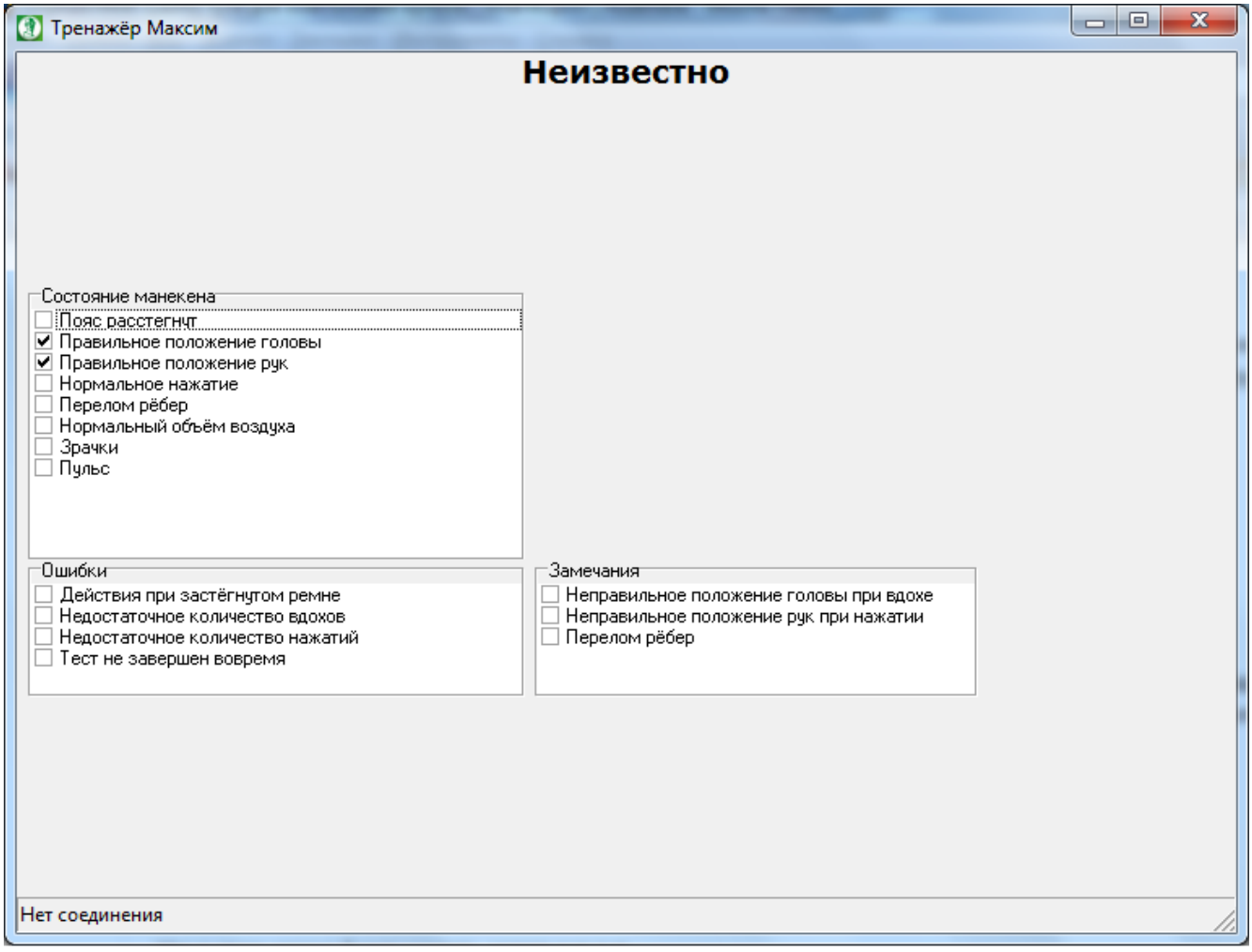
1. Приложение должно иметь максимально понятный и удобный интерфейс
2. Приложение должно иметь низкие системные требования
3. Приложение должно наглядно показывать процесс сердечно-лёгочной реанимации.

Помимо этого были наложены ограничения, связанные со спецификой данного тренажёра:

1. Соединение с тренажёром идёт через COM-порт
2. Приложение поддерживает два режима – обучающий и тестовый
3. Всё управление осуществляется с пульта, входящего в комплект тренажёра, через который последний соединяется с компьютером.

Обзор

Когда была поставлена задача реализовать данное приложение, была передана текущая версия программного обеспечения для тренажёра «Максим» (рис. 1). В ней полностью отсутствовал пользовательский интерфейс, не было визуализации и отражения текущих действий. По сути это был пульт для компьютера, на котором загорались индикаторы, и, было очевидно, что медицинскому работнику будет достаточно сложно разобраться в таком приложении. Ниже представлено, как оно тогда выглядело.



(Рис. 1)

Формальная постановка задачи

В результате была поставлена следующая задача: разработать и реализовать приложение для тренажёра сердечно-лёгочной реанимации «Максим». Общение с тренажёром происходит через COM-порт по заданному производителем протоколу. Управление приложением происходит по средствам сигналов, приходящих с тренажёра, в ответ на запрос его состояния, само приложение не должно менять состояния тренажёра. Приложение должно работать в двух режимах – тестовом и учебном, при этом тестовый режим может быть запущен в нескольких вариантах: 1-5, 2-15, 2-30 (первое число — количество вдохов, второе число — количество нажатий), а также 30-2 (то же, что и 2-30, но тест начинается с нажатий). Помимо этого к интерфейсу приложения были предъявлены следующие требования:

1. Учебный режим
    a. Явно отображать те действия, которые обучающийся должен будет проделывать с тренажёром
    b. Должен присутствовать звуковой комментарий
    c. Обучающийся должен иметь возможность попробовать выполнить с тренажёром заданные действия, при этом он должен увидеть в приложении, как реагирует на его действия тренажёр (так, например, при нажатии на грудную клетку должны сужаться глаза, сжиматься сердце, а кровь должна пробегать круг по телу).
2. Тестовый режим
    a. Обучающийся должен видеть диаграмму, которая будет отражать его действия.
    b. Обучающийся должен видеть индикатор ритма, индикатор биения сердца и индикатор реакции зрачка
    c. Обучающийся должен видеть список допущенных им ошибок с указанием времени.
    d. Тест идёт 1 минуту, в названии теста указывается режим, по окончании выводится результат.

е. После завершения теста можно сохранить диаграмму с результатом в виде картинки, а также распечатать её.

Решение задачи

Выбор средств

Приложение было реализовано на языке C# на платформе .NET 2.0. Выбор пал именно на эту среду по нескольким причинам. Во-первых, C# - это язык высокого уровня, разработка на нём на порядок проще, чем на C/C++, он имеет все необходимые возможности, а также содержит огромное количество библиотек, которые позволяют не реализовывать многие вещи самостоятельно, а переиспользовать готовые решения. Версия 2.0 была выбрана для уменьшения требований приложения по предустановленному программному обеспечению, и так как её возможностей полностью хватало для выполнения поставленной задачи. Также на C# удобно и просто делать GUI, а среда Visual Studio 2008 доступна студентам по программе MSDNAA. Альтернативным языком программирования была JAVA, но, к сожалению, среда разработки IntelliJ Idea платная, а NetBeans недостаточно удобен для создания GUI, как и Eclipse, что было важно, так как на реализацию данного продукта были отведены короткие сроки.

Приложение было написано с использованием технологий WinFroms и Thread, как наиболее простых и достаточных для реализации проекта в кратчайшие сроки. Разработка шла в IDE Visual Studio 2008.

Архитектура приложения

В проекте я занимался разработкой архитектуры приложения, а также реализацией нижнего уровня взаимодействия через COM-порт, сменой состояний и учебным режимом.

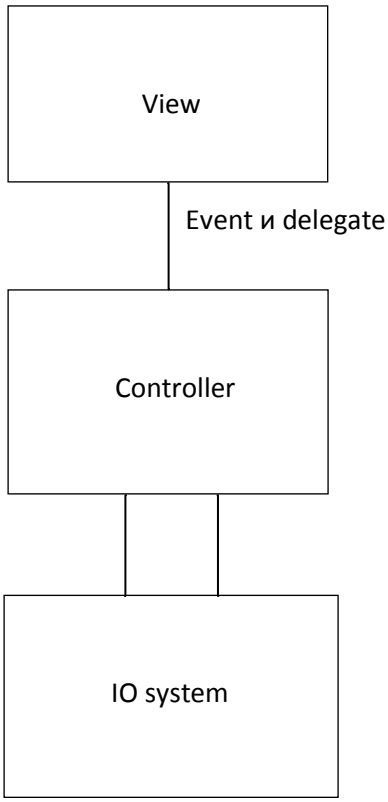
Для архитектуры приложения был выбран паттерн MVC[2]. Единственное, часть Model была сокращена до минимума. Основными частями стали View и Controller. Они оформлены в виде двух потоков:

1. Controller – поток обработки сообщений, на который были возложены обязанности по отправке и принятию сообщений через COM-порт, их разбор в соответствии с протоколом, создание представления в виде классов C# и отправка событий во View об изменении состояния тренажёра.

2. View – главный поток приложения, содержащий в себе форму. Отвечает за визуализацию всего, что происходит на тренажёре. Слушает события от Controller’а и производит соответствующие изменения в интерфейсе.

Изначально приложение было оформлено следующим образом. При запуске создаётся основной поток приложения, потом создаётся поток Controller, после чего, основной поток принимает от него сообщение о режиме, включенном на тренажёре. В зависимости от режима запускается поток View в учебном или тестовом варианте соответственно. При смене режима на тренажёре поток View останавливается и запускается в новом варианте. Но при таком решении возникла проблема утечки памяти. При частой смене режимов, что приводило к частой смене потоков, не всегда уничтожались объекты, созданные потоком, что приводило к тому, что через некоторое время приложение съедало всю имеющуюся оперативную память. Для решения этой проблемы была предпринята попытка вручную уничтожать все ссылки на создающиеся объекты, чтобы потом успешно собрать их Garbage Collector'ом. Но полностью осуществить эту идею не удалось, так как возникла следующая ситуация: поток создаётся и вызывает конструктор формы, в этот момент потоку приходит команда завершиться, но он ещё не создался до конца. В результате чего поток завершался, но оставлял в памяти некоторое количество мусора, который было не отследить и не удалить.

Поэтому пришлось немного изменить архитектуру части View и сделать её в виде одного потока. А именно: раньше было две формы: одна - для тестового режима, вторая - для учебного. Каждая из них запускалась в отдельном потоке, из-за чего и происходила утечка памяти. Теперь каждая форма была превращена в панель, и просто переключалась видимость панелей, что, правда, привело к большим затратам памяти единовременно, но даже при таком подходе расход был в пределах нормы. Это дало возможность мгновенного переключения между режимами, мгновенного и безболезненного прерывания любого из режимов, а также упрощению архитектуры. Теперь при старте приложения запускался основной поток – поток формы, а потом запускался поток Controller. После чего приложения начинало функционировать как и ранее, за исключением того, что при смене режима одна панель просто становилась невидимой, а другая видимой. В результате общая архитектура приложения стала такой, как изображено на рис. 2.



(Рис. 2)

Общение с тренажёром

Была решена проблема с памятью, но помимо неё нужно было решить такую задачу: тренажёр сообщает своё состояние только в ответ на запрос. Поэтому, очевидно, тренажёр всегда надо опрашивать, для чего и создан поток Controller. Но как часто надо посылать запросы, чтобы не загружать систему и при этом не пропустить важное событие? Ответ на этот вопрос подбирался эмпирическим путём. В результате было выявлено, что нужна задержка примерно в 5 мс.

При решении данной задачи ещё возникла следующая проблема: ответ на запрос тренажёр присылался не целиком, а по частям, байт за байтом, в течение некоторого, заранее не известного, интервала времени. При этом если в этот момент ему отправить ещё один запрос, то он вначале пришлет ответ на предыдущий, а после чего начнёт присылать ответ на новый. В результате чего изначально иногда ответы начинали путаться. Это удалось решить довольно легко – был просто увеличен тайм аут, который отводился на приём одного ответа. Так как каждый из ответов по протоколу начинался и заканчивался специальным байтом, то можно было идентифицировать начало и конец сообщения, потому после прихода байта начала в течение некоторого времени из входного буфера считывались байты. Если

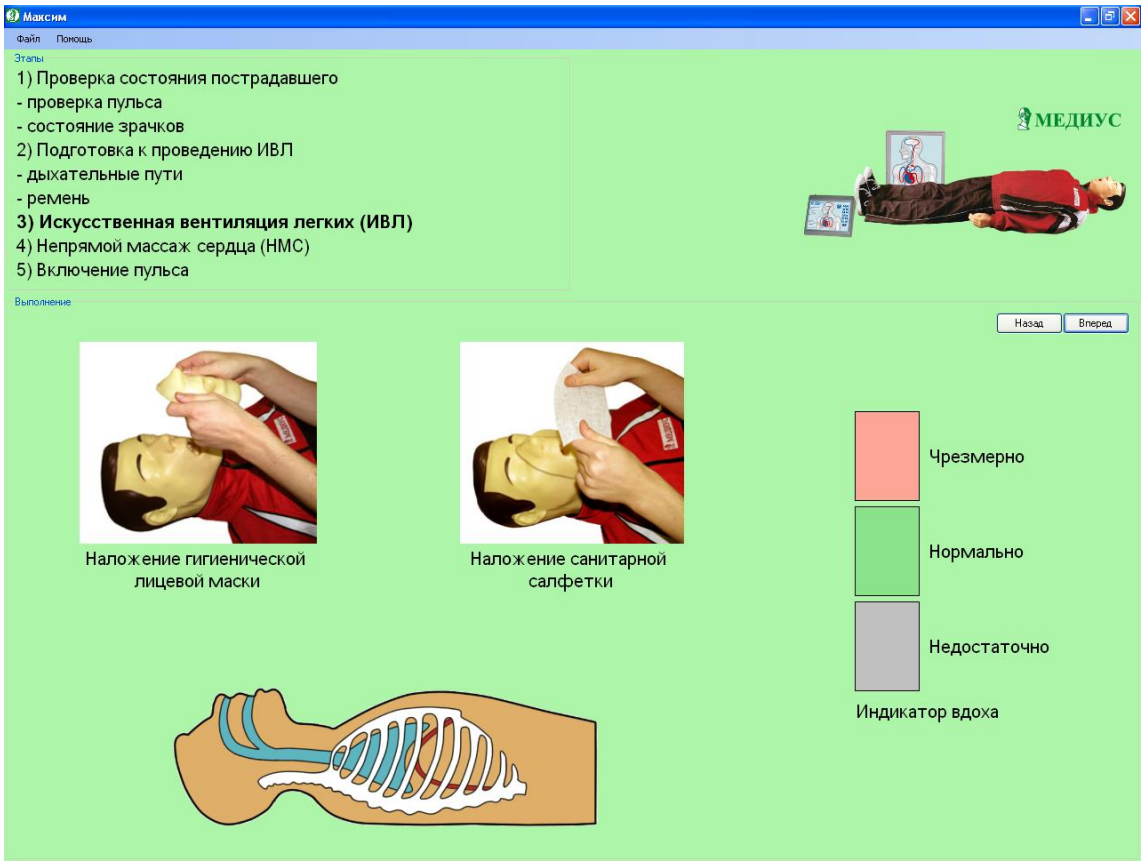
встречался байт конца, то считывание прекращалось, и ответ передавался на более высокий уровень приложения. Если по прошествии времени байт конца так и не приходил, то программа считала, что тренажёр не получил запрос и отправляла его вновь. За счёт увеличения тайм аута удалось избавиться от наложения старого ответа на новый запрос и программа начала функционировать корректно.

Но этими трудности с COM-портом не ограничились. Следующей стало падение программы при отключении тренажёра от компьютера во время работы приложения. Вначале долго не удавалось установить, из-за чего происходит такая ошибка. Первые идеи были обернуть работу с портом в try – catch и ловить ошибку, происходящую при попытке чтения из закрытого порта, но это не помогало. Следующее, что было предпринято – добавить обёртку на более высоких уровнях, так как возможно исключение происходило не в самой работе порта, а выше. Это тоже не помогло, но было установлено, что исключение, которое мы хотели ловить, ловится и не является причиной падения приложения. Для исправления данной ошибки пришлось обратиться к научному руководителю. Вместе мы выяснили, что ошибка происходит в функции dispose() библиотеки .NET, и что данное исключение невозможно обработать средствами C#, что является ошибкой платформы, так как по спецификации .NET гарантируется, что в этой функции не может происходить исключений.

Заключение

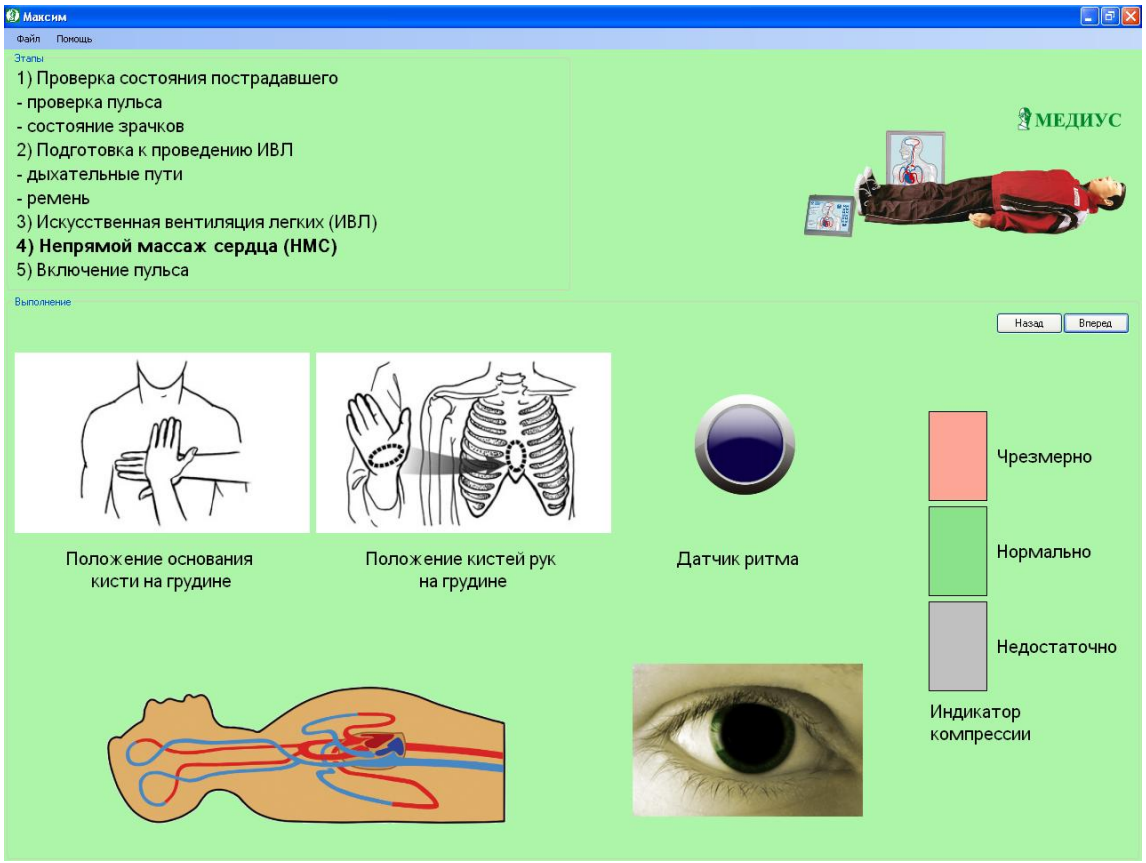
В результате работы было получено обучающее приложение для тренажёра «Максим». Были реализованы два режима тестовый и учебный, при этом приложение могло свободно переключаться между ними. Переключение происходит практически мгновенно, что важно, так как управление режимами осуществляется не самой программой, а внешним устройством, поэтому любая задержка может привести к тому, что мы пропустим какое-то важное изменение в состоянии тренажёра, и программа будет работать некорректно. Также удалось добиться полной наглядности в интерфейсе, а именно создать анимации, которые двигаются в зависимости от датчиков на тренажёре и корректно отображают то, что происходит с тренажёром и человеком при сердечно-лёгочной реанимации. Помимо этого я научился работать с устройствами, соединяемыми по COM-порту и имеющими протокол со всеми значащими битами, освоил модель MVC и построение приложения на основе событийной модели, когда на любое происходящее изменение в состоянии источника шлётся событие, содержащее уже данные, необходимые для части представления.

Теперь я хочу привести скриншоты приложения.



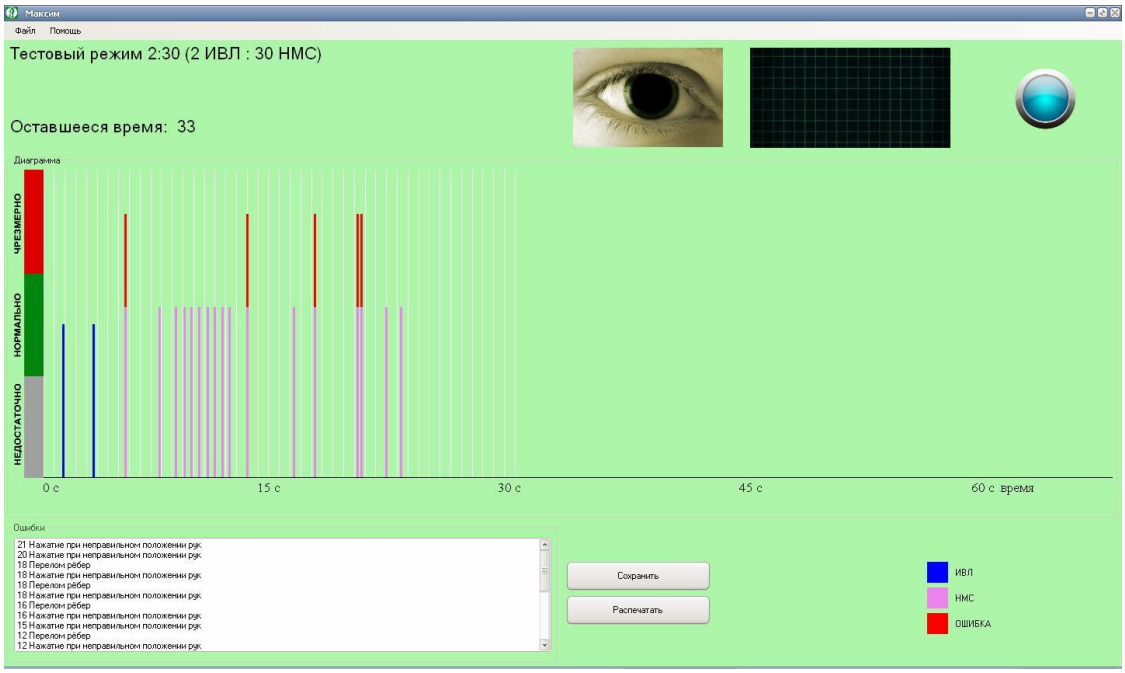
(Рис. 3)

Для начала, учебный режим. Как видно, интерфейс оснащен датчиками (рис. 3), которые показывают силу вдоха (вертикальный из трёх полос), поведение грудной клетки, при вдохе (снизу), также мы видим этап, на котором мы находимся (слева вверху). Все датчики полностью синхронизированы с тренажёром. При этом если в любом из состояний неожиданно будет переключен режим, в котором работает тренажёр (а это возможно), то приложение поведёт себя корректно и сразу перейдёт в необходимый.



(Рис. 4)

На рис. 4 видим датчик силы нажатия, индикатор поведения глаза, который отображает поведение глаза человека при реанимации, а также слева внизу можно наблюдать разрез человека, на котором видно, как движется кровь при реанимации.



(Рис. 5)

На рис. 5 представлен уже тестовый режим. Как и требовалось, имеется диаграмма, на которой отображаются разными цветами действия испытуемого, а также их сила (красный цвет показывает, что действие было чрезмерным). Также есть бледные вертикальные полосы, для более удобной привязки к временной шкале. Имеется ряд датчиков, которые полностью синхронизированы с тренажёром и позволяют сразу видеть результат своих действий. Все ошибки выводятся слева снизу.

Список литературы

[1]Описание тренажёра «Максим III-01»//Сайт завода производителя URL: http://www.medius.ru/ (дата обращения: 27.05.2010)

[2]MVC//Википедия – всемирная энциклопедия URL: http://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 27.05.2010)

Справочная информация по C#//Библиотека MSDN URL: http://msdn.microsoft.com/ru-ru/library/ms123401.aspx (дата обращения: 27.05.2010)

Справочная информация по C#//Форум .net разработчиков gotdotnet URL: http://www.gotdotnet.ru/forums/ (дата обращения: 27.05.2010)