**РЕФЕРАТ**

Задачей данной работы была разработка трехмерной графической модели проводящей системы сердца.

С этой целью была изучена специфическая для данной области литература, изучен язык программирования шейдеров, а также изучен метод, основанный на использовании предразмеченных карт распространения сигнала.

С помощью шейдеров была создана модель миокарда предсердия, имитирующая распространение возбуждения по поверхности сердечной мышцы.

Кроме того, в работе приводится подробное описание модели и использованного средства разработки. Это позволяет использовать работу в учебных целях, а также она может использоваться специалистами при моделировании как нормальных, так и патологических работах сердца.

**СОДЕРЖАНИЕ**

Реферат………………………………………………………….…………………2

Введение………………………………………………………………….………..4

Глава 1. Проводящая система сердца……………………………...…………….6

Глава 2. Описание модели…………………………………......…….………….11

2.1. Методы решения проблемы…………………………….………….12

2.2. Теоретическое описание алгоритма……………………………….16

Глава 3. Шейдеры и возможности программирования………………………..19

Глава 4. Реализация алгоритма……………………………………..…………..22

4.1. Результаты экспериментальных запусков…………….…………..27

Заключение…………………………………………………………….…………33

Список используемых источников……………………………….….…………34

Приложение 1…………………………………………………………………….37

**ВВЕДЕНИЕ**

Сердце является одним из главных органов человеческого организма,

поскольку выполняет очень важную функцию – оно является мотором, прогоняющим кровь по всему организму. Кровь в свою очередь несет в себе различные вещества (кислород, питательные вещества), без которых само существование цельного человеческого организма в том виде, в котором он существует, было бы невозможно. Помимо «механических» функций сердце имеет еще и функции по обеспечению ритмичной работы, функции по управлению процессом «перекачки» крови в организме.

К сожалению, сердце, как и любой орган человеческого организма подвержен заболеваниям и потрясениям. Однако, учитывая особый статус сердца как мотора организма, эти нарушения могут быть чрезвычайно опасными.

Сейчас в области медицины (сердца) проводится множество исследований, для которых могут быть полезны компьютерное моделирование и компьютерная графика. Компьютерная визуализация уже широко применяется для диагностики, планирования и проведения операций. Также создаются модели, которые имитируют работу различных биологических систем, в частности сердца.

Цикл компьютерного моделирования содержит три основные стадии:

- подготовка к анализу,

- вычисления,

- визуализация и анализ результатов.

В полном виде этот цикл (известный также как цикл Самарского, схема численного моделирования или схема численного эксперимента) предусматривает следующие стадии:

- сбор и накопление первичных данных;

- разработка физической модели;

- разработка математической модели;

- алгоритмизация;

- программирование;

- вычисление по программе;

- визуализация;

-интерпретация и анализ результатов.

Таким образом, можно говорить о моделях разного уровня, как о физических, математических, численных и даже графических и/или визуальных моделях. В рамках нашей работы мы разрабатываем трехмерную графическую модель проводящей системы сердца, поэтому необходимо заострить внимание именно на двух последних стадиях цикла Самарского, а именно: визуализации, интерпретации и анализе результатов.

**1. ПРОВОДЯЩАЯ СИСТЕМА СЕРДЦА**

Знание проводящей системы сердца необходимо для освоения ЭКГ и понимания сердечных аритмий.

Сердце обладает автоматизмом — способностью самостоятельно сокращаться через определенные промежутки времени. Это становится возможным благодаря возникновению электрических импульсов в самом сердце. Оно продолжает биться при перерезке всех нервов, которые к нему подходят.

Импульсы возникают и проводятся по сердцу с помощью так называемой проводящей системы сердца. Рассмотрим компоненты проводящей системы сердца:

* синусно-предсердный узел,
* предсердно-желудочковый узел,
* пучок Гиса с его левой и правой ножкой,
* волокна Пуркинье.

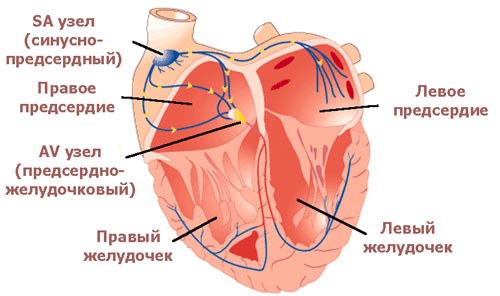


Рис.1 Схема проводящей системы сердца

Теперь рассмотрим каждый компонент в отдельности.

1) синусно-предсердный узел (синусовый, синоатриальный, SA; от лат. *atrium* - предсердие) — источник возникновения электрических импульсов в норме. Именно здесь импульсы возникают и отсюда распространяются по сердцу. Cинусно-предсердный узел расположен в верхней части правого предсердия, между местом впадения верхней и нижней полой вены. Слово “синус” в переводе означает “пазуха”, “полость”.

Фраза “синусовый ритм” в расшифровке ЭКГ означает, что импульсы генерируются в правильном месте — синусно-предсердном узле. Нормальная частота ритма в покое — от 60 до 80 ударов в минуту. Частота сердечных сокращений (ЧСС) ниже 60 в минуту называется брадикардией, а выше 90 — тахикардия. У тренированных людей, таких как спортсмены, наблюдается брадикардия.

Известно, что в норме импульсы генерируются не с идеальной точностью. Существует дыхательная синусовая аритмия (ритм называется неправильным, если временной интервал между отдельными сокращениями на ≥ 10% превышает среднее значение). При дыхательной аритмии ЧСС на вдохе увеличивается, а на выдохе уменьшается, что связано с изменением тонуса блуждающего нерва и изменением кровенаполнения отделов сердца при повышении и понижении давления в грудной клетке. Как правило, дыхательная синусовая аритмия сочетается с синусовой брадикардией и исчезает при задержке дыхания и увеличении ЧСС. Дыхательная синусовая аритмия бывает преимущественно у здоровых людей, особенно молодых. Появление такой аритмии у лиц, выздоравливающих после инфаркта миокарда, миокардита и др., является благоприятным признаком и указывает на улучшение функционального состояния миокарда.

2) предсердно-желудочковый узел (атриовентрикулярный, AV; от лат. *ventriculus* — желудочек) является, можно сказать, “фильтром” для импульсов из предсердий. Он расположен возле самой перегородки между предсердиями и желудочками. В AV-узле самая низкая скорость распространения электрических импульсов во всей проводящей системе сердца. Она равна примерно 10 см/с (для сравнения: в предсердиях и пучке Гиса импульс распространяется со скоростью 1 м/с, по ножкам пучка Гиса и всем нижележащим отделам вплоть до миокарда желудочков — 3-5 м/с). Задержка импульса в AV-узле составляет около 0.08 с, она необходима, чтобы предсердия успели сократиться и перекачать кровь в желудочки.

Есть аритмии, при которых нарушается формирование и распространение импульсов в предсердиях. Например, при мерцательной аритмии (фибрилляция предсердий) волны возбуждения беспорядочно циркулируют по предсердиям, но AV-узел блокирует большинство импульсов, не давая желудочкам сокращаться слишком часто. С помощью различных препаратов можно регулировать ЧСС, повышая проводимость в AV-узле (адреналин, атропин) или снижая ее (дигоксин, верапамил, бета-блокаторы). Постоянная мерцательная аритмия бывает тахисистолической (ЧСС > 90), нормосистолической (ЧСС от 60 до 90) или брадисистолической формы (ЧСС < 60). Это одна из самых частых аритмий, ею страдает > 6% больных старше 60 лет. Известно, что с фибрилляцией предсердий жить можно годами, а вот фибрилляция желудочков является смертельной аритмией , при ней без экстренной медицинской помощи больной умирает за 6 минут.



Рис. 2 Проводящая система сердца

3) Пучок Гиса (предсердно-желудочковый пучок) не имеет четкой границы с AV-узлом, проходит в межжелудочковой перегородке и имеет длину 2 см, после чего делится на левую и правую ножки соответственно к левому и правому желудочку. Поскольку левый желудочек работает интенсивнее и больше по размерам, то левая ножка делится на две ветви — переднюю и заднюю.

Патологические процессы (некроз, воспаление) могут нарушать распространение импульса по ножкам и ветвям пучка Гиса, что видно на ЭКГ. В таких случаях в заключении ЭКГ пишут, например, “полная блокада левой ножки пучка Гиса”.

4) Волокна Пуркинье связывают конечные разветвления ножек и ветвей пучка Гиса с сократительным миокардом желудочков.

Способностью генерировать электрические импульсы (т.е. автоматизмом) обладает не только синусовый узел. Природа позаботилась о надежном резервировании этой функции. Синусовый узел является водителем ритма первого порядка и генерирует импульсы в частотой 60-80 в минуту. Если по какой-то причине синусовый узел выйдет из строя, станет активным AV-узел — водитель ритма 2-го порядка, генерирующий импульсы 40-60 раз в минуту. Водителем ритма третьего порядка являются ножки и ветви пучка Гиса, а также волокна Пуркинье. Автоматизм водителя ритма третьего порядка равен 15-40 импульсов в минуту. Водитель ритма также называют пейсмекером (pacemaker, от англ. *pace* — скорость, темп).

В норме активен только водитель ритма первого порядка, остальные “спят”. Это происходит, потому что электрический импульс приходит к другим автоматическим водителям ритма раньше, чем в них успевает сгенерироваться собственный. Если автоматические центры не повреждены, то нижележащий центр становится источником сокращений сердца только при патологическом повышении его автоматизма (например, при пароксизмальной желудочковой тахикардии в желудочках возникает патологический источник постоянной импульсации, которая заставляет миокард желудочков сокращаться в своем ритме с частотой 140-220 в минуту).

**2. ОПИСАНИЕ МОДЕЛИ**

В рамках нашей работы мы разрабатываем трехмерную графическую модель проводящей системы сердца. Важность разработки такого рода систем возрастает пропорционально росту объёмов информации, требуемой в современных научных исследованиях, и скорости, с которой пользователь данных систем должен получать необходимые для него данные в удобной для восприятия форме.

Развитие современных компьютерных технологий позволяет современным графическим системам удовлетворять таким требованиям, как быстрая обработка данных, хранение больших объёмов информации, качественное графическое представление данных и многое другое, однако вопрос о том, как именно должна быть представлена та или иная информация, остаётся открытым и является одним из ключевых вопросов в компьютерной визуализации.

Возникает также и проблема в том, что в математическом моделировании существует множество стандартных подходов к созданию образов для отображения различных информационных объектов, в то время как эти образы могут оказаться вовсе не привычными и не приемлемыми для областей, не связанных напрямую с математикой. Поэтому иногда при создании таких графических систем необходимо генерировать некие новые методы для представления информации.

Как было сказано выше, сердце обладает способностью сокращаться благодаря возникающим в самом сердце электрическим импульсам. В здоровом сердце источником таких импульсов является синусно-предсердный узел, расположенный в верхней части правого предсердия.

**2.1. Методы решения проблемы**

В данной работе возникает следующая задача: построить модель, представляющую собой модель миокарда предсердия и имитирующую распространение возбуждения по поверхности сердечной мышцы. Поверхность представлена в виде множества связанных между собой клеток, которые могут передавать друг другу сигнал.

Система будет реализована с помощью компьютерной графики и компьютер здесь является основной базой для самого процесса моделирования, поэтому эту модель можно назвать качественной компьютерной моделью.

Для решения данной задачи существует несколько методов:

1. поиск в ширину/ поиск в глубину;

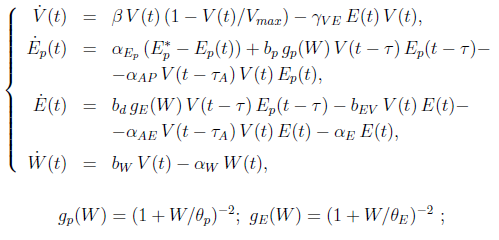
2. построение и решение уравнения с запаздыванием в каждой точке;

3. использование предразмеченных карт распространения сигнала.

Рассмотрим преимущества и недостатки каждого метода:

1. Подходы к решению задачи, основанные на поиске в ширину или в глубину достаточно просты в реализации. Они довольно точно позволяют строить карту распространения сигнала. Но есть несколько существенных недостатков данных методов, которые препятствуют реализации поставленной задачи, а именно: эти подходы не позволяют строить более одной итерации распространения волны (в случае наличия областей медленной проводимости это является критичным моментом), а также не позволяют динамически менять карту проходимости сигнала по миокарду.
2. Подход, основанный на построении и решении уравнения с запаздыванием, в отличие от первого сложен в реализации. В то же время, благодаря этому методу, можно динамически оценивать полное состояние системы в каждый момент времени. Все алгоритмы имеют замкнутую математическую модель, которая позволяет все асимптотически оценить. Точное решение уравнения с запаздыванием позволяет доказать все результаты. Несмотря на плюсы данного метода, есть существенный недостаток: дифференциальные уравнения, описывающие реальные процессы, достаточно сложно построить и в общем случае они не имеют решения.

Например, LCMV- модель: уравнения модели описывают скорость изменения плотности популяции вирусов *V(t)*, CTL-предшественников – *Ep(t).* CTL-эффекторов – *E(t)* и уровень вирусной нагрузки *W(t):*



*β* – константа скорости репликации вирусов;

*Vmax* – максимальная вирусная концентрация в селезенке;

*γVE* – константа нейтрализации вирусов за счет действия эффекторов;

*αEp* – константа скорости естественной гибели для CTL-предшественников;

*Ep\** – гомеостатическая концентрация CTL-предшественников в селезенке до заражения;

*bp* – константа скорости стимуляции CTL-предшественников;

*θp* – уровень вирусной нагрузки, вызывающий снижение скорости деления CTL-предшественников в 2 раза;

*τ* – продолжительность цикла деления CTL-предшественников;

*αAP* – константа скорости апоптоза для CTL-предшественников;

*τА* – продолжительность процесса формирования вирус- индуцированного апоптоза у CTL-предшественников;

*bd* – константа скорости диффенциации CTL-предшественников;

*θЕ* – уровень вирусной нагрузки, вызывающий снижение в 2 раза скорости дифференцировки CTL-предшественников в эффекторы;

*bEV* – константа скорости гибели CTL-эффекторов вследствие взаимодействий с инфицированными клетками;

*αАЕ* – константа скорости апоптоза для CTL-эффекторов;

*αЕ* – константа скорости естественной гибели для CTL-эффекторов;

*bW* – константа скорости увеличения вирусной нагрузки;

*αW* – константа скорости уменьшения подавляющего эффекта вирусной нагрузки.

Модель была исследована, используя численную технологию бифуркационного анализа для функционально- дифференциальных уравнений[6].

1. Метод, основанный на использовании предразмеченных карт распространения сигнала, как и метод, основанный на поиске в ширину или в глубину, не предназначен динамически изменять прохождение сигнала по поверхности модели сердца, но в отличие от него может использовать несколько предразмеченных карт. Это обусловлено тем, что алгоритмы данного метода занимают мало места в памяти. Также использование предразмеченных карт распространения сигнала имеет высокое быстродействие и низкую вычислительную сложность. Немаловажным является то, что нужно предварительно получить такую карту распространения сигнала, это можно сделать разными способами:

- построить на основе поиска в ширину/в глубину;

- получить из реальных медицинских наблюдений [7];

- из данных численного решения математических моделей [8];

- получить из данных, предоставленных методом медицинского диагностирования.

При решении поставленной задачи будет использован третий метод, так как он имеет большие преимущества и несущественные недостатки, которые можно разрешить в ходе решения задачи.

**2.2. Теоретическое описание алгоритма**

Предразмеченная карта распространения сигнала- это размеченное множество точек вида {(x,y),t} (x,y)∊ℕ t∊[0,1), в котором (x,y)- пространственные координаты точки, t- момент времени, в который сигнал дойдет до точки (x,y), то есть клетка сердца будет активна.

Будем считать, что описанная выше карта у нас есть. Необходимо ее визуализировать с учетом следующих требований: в работе рассматривается лишь электрическая модель сердца, то есть здесь не рассматриваются механические и химико-биологические процессы, происходящие в миокарде. Предполагается, что электрический сигнал, распространяющийся по миокарду, имеет всего два уровня – либо сигнал есть, либо его нет, то есть не учитывается фактический уровень идущего сигнала, который реально может принимать все промежуточные значения. Кроме того, условно считается, что миокард однослойный, что несколько упрощает построение модели, так как клетка передаёт сигнал всем соседним с ней клеткам, и рассмотрение многослойного варианта существенно увеличило бы сложность модели.

Взаимодействие между клетками выражается в передаче сигнала. Каждая клетка находится в одном из трёх состояний: в невозбуждённом состоянии, в состоянии активности и в рефрактерном состоянии, каждое из которых далее описано более подробно.

Невозбуждённое состояние – это свободное состояние клетки, при котором клетка полностью восприимчива к поступающим к ней сигналам, но сама пока не обладает достаточно высоким электрическим потенциалом, который характеризует возбуждённое, активное состояние клетки. Изначально все клетки находятся в невозбуждённом состоянии. При этом одна или группа клеток объявляется генератором сигнала. С медицинской точки зрения, это либо основной источник электрического сигнала в сердце – синусовый узел, либо дополнительные патологические генераторы сигнала, называемые экстрасистолами. Эти клетки-генераторы посылают сигнал другим клеткам. Схема взаимодействия следующая: получившая сигнал клетка передаёт его с некоторой задержкой всем соседним клеткам, находящимся в невозбуждённом состоянии. Клетку, только что получившую сигнал, будем называть активной. При этом клетка обладает достаточным электрическим потенциалом, чтобы возбудить соседние с ней клетки, находящиеся в невозбуждённом состоянии. Активность клетки определяется конкретным временным периодом. По окончании периода активности клетка передаёт сигнал соседним с ней клеткам и переходит в следующую фазу, называемую рефрактерной, в течение которой клетка становится невозбудимой к приходящим к ней сигналам. В рефрактерном состоянии клетка не возбуждена, то есть она не обладает достаточно высоким электрическим потенциалом, чтобы возбудить соседние клетки.

В данной модели допущено ограничение: мы считаем, что никакой вообще сигнал не способен возбудить клетку в рефрактерном состоянии. На самом же деле клетки реального миокарда могут находиться также и в состоянии относительной рефрактерности, где основным критерием возбудимости служит уровень поступающих к клетке сигналов. По прохождении периода рефрактерности клетка снова входит в начальное, невозбуждённое состояние и ожидает прихода нового сигнала.

Следует отметить, что упор делается на качественное описание объекта и что это именно компьютерное, а не чисто математическое моделирование. В данном случае, моделирование опирается на свойства объекта, которые были выявлены медиками эмпирически, а не на основы некоторой математической модели, так как, строго говоря, такой глобальной модели, описывающей работу всего миокарда, пока не существует.

Управление моделью – также целиком качественное воздействие, мы не задаём каких-либо пропорций, соотношений или законов взаимодействия, а влияем непосредственно на поведение отдельных её частей, что указывает скорее на физический подход к изучению свойств объекта исследования, свойственный естественным наукам и, в частности, медицине.

**3. ШЕЙДЕРЫ И ВОЗМОЖНОСТИ ПРОГРАММИРОВАНИЯ**

Современные графические процессоры (GPU- Graphics Processing Unit) очень эффективно обрабатывают и отображают компьютерную графику. Благодаря специализированной конвейерной архитектуре они намного эффективнее в обработке графической информации, чем типичный центральный процессор (CPU).

Высокая вычислительная мощность GPU объясняется особенностями архитектуры: графический процессор изначально создавался как многоядерная структура, в которой количество ядер может достигать сотен. В отличие от CPU, GPU исторически предназначался для обработки компьютерной графики, поэтому рассчитан на массивно параллельные вычисления.

Программирование графических процессоров прошло такой же путь развития, как и для центральных процессоров. Не существовало способов эффективно писать программы для первых графических процессоров. Если разработчик приложения хотел включить специальные эффекты для обработки изображения, то ему приходилось либо добавлять обработчики, работающие на CPU, что значительно замедляло скорость выполнения программы, либо довольствоваться сильно ограниченным набором поддерживаемых аппаратно эффектов.

Для удобного написания программ разработчикам требовалось создать специализированный язык, как в свое время для CPU был разработан ассемблер, который бы позволял создавать простые и быстрые программы (язык должен быть приближен к возможностям аппаратной части) для создания и применения различных эффектов. Была разработана и стандартизирована концепция шейдеров.

Ше́йдер (англ. Shader) — это программа, используемая в трёхмерной графике для определения окончательных параметров объекта или изображения. Она может включать в себя произвольной сложности описание поглощения и рассеяния света, наложения текстуры, отражение и преломление, затемнение, смещение поверхности и эффекты пост-обработки.

По своему типу шейдеры делятся на два основных типа:

• вершинные шейдеры (vertex shaders) оперируют с различными геометрическими данными вершин объектов в трехмерном пространстве;

• пиксельные или фрагментные шейдеры (pixel / fragment shaders) работают с фрагментами изображения, пикселями, преобразуя их некоторым образом, что может использоваться для постпроцессинга (post-processing) изображения – преобразования уже полученного изображения в новое (например, для применения эффекта черно-белого цвета).

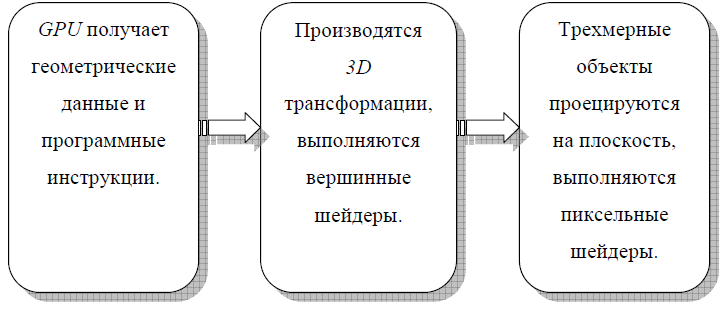


Рис. 3 Упрощенная схема графического конвейера

Как показано на рисунке выше, сначала в GPU происходит получение программных инструкций и геометрических данных. Затем после 3D трансформаций идет выполнение вершинных шейдеров. На последнем этапе происходит проецирование трехмерных объектов на плоскость. На объекты накладывается текстура (двумерное изображение, накладываемое на грани трехмерных объектов), а для каждого пикселя вычисляется результирующее значение цвета. Далее для каждого пикселя выполняются фрагментные шейдеры. В результате работы последнего из них получается финальный цвет пикселя, который и выводится на экран. Таким образом, каждый фрагментный шейдер будет выполнен N раз, где N – число пикселей результирующего изображения. Именно за счет наличия множества конвейерных процессоров фрагментные шейдеры выполняются быстро.

Основная структура программы на языке шейдеров аналогична структуре C-программы. Точкой входа группы шейдеров является функция, тело которой находится внутри фигурных скобок. Константы, идентификаторы, операторы, выражения и предложения — понятия, идентичные для языка шейдеров и языка C. Организация потока в циклах, if–then–else и вызовы функций в языке шейдеров практически такие же, как в C.

Цель языка шейдеров— дать приложениям возможность определять, как будет происходить обработка графики на некоторых этапах с помощью высокоуровневого языка программирования, сконструированного специально для этого.

Языков программирования шейдеров существует целых 4: ассемблер, HLSL (High Level Shader Language, от Microsoft), GLSL (OpenGL Shader Language, от SGI) и Cg от NVidia.

- HLSL используется в Direct3D, синтаксис очень близок к C (не С++);

- GLSL используется в OpenGL-приложениях, по синтаксису похож на HLSL;

- Cg повторяет синтаксис HLSL, и является промежуточным языком, который автоматически компилируется в HLSL или GLSL, в зависимости от того, с каким приложением мы работаем: OpenGL или Direct3D.

**4. РЕАЛИЗАЦИЯ АЛГОРИТМА**

Решение задачи реализовано на Unity3D (мультиплатформенный инструмент для разработки двух- и трёхмерных приложений, работающий под операционными системами Windows, Linux и OS X).

Вид при первом запуске:

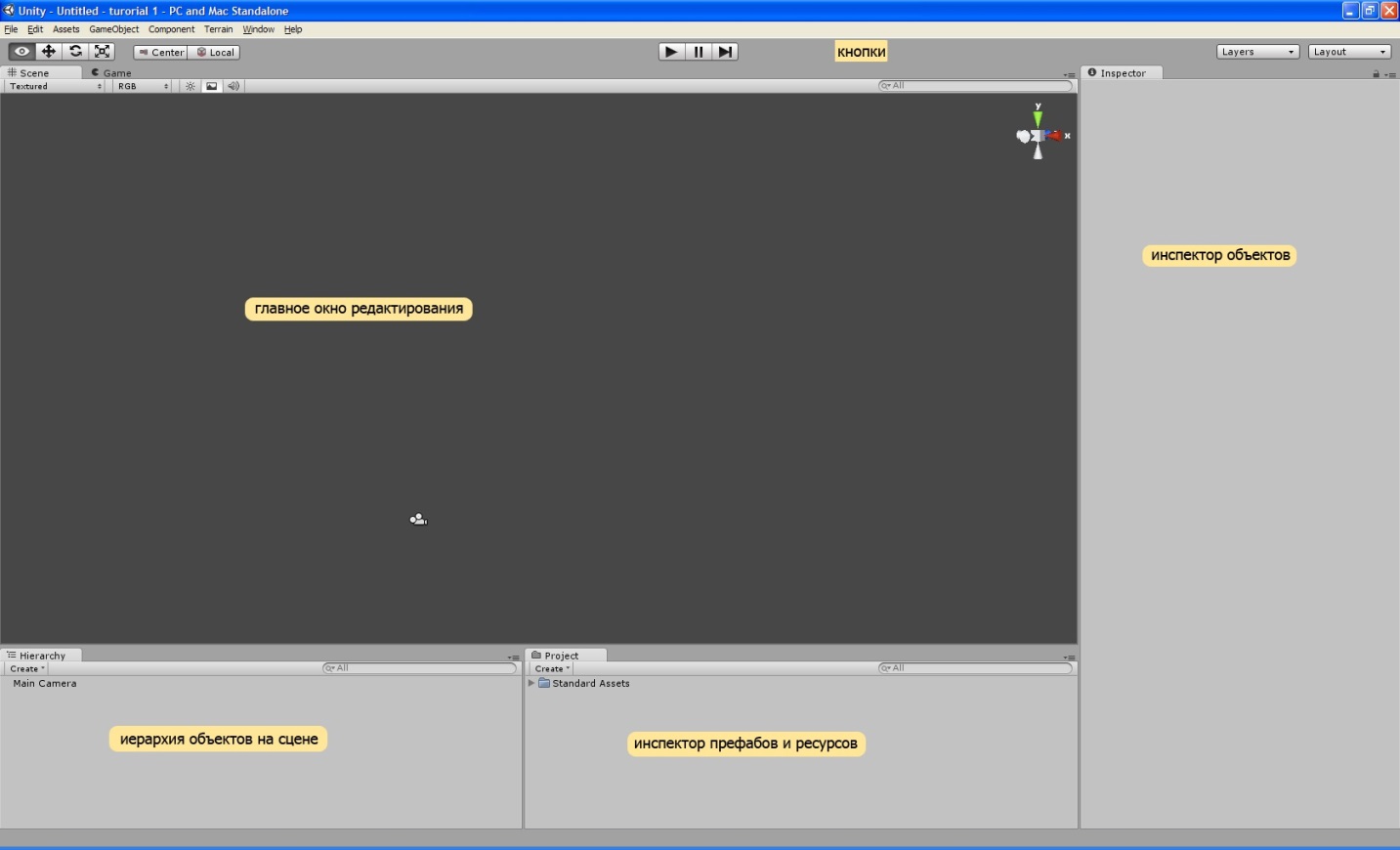


Рис. 4 Интерфейс средства разработки

- инспектор префабов и ресурсов — это место, где хранятся добавленные в проект модели, текстуры, звуки, и префабы — сохранённые для дальнейшего повторного использования объекты;

- иерархия объектов на сцене — это список всех объектов на текущем уровне, показывающий так же отношения Parent-Child;

- инспектор объектов показывает компоненты и их свойства выделенного в данный момент объекта — модели, текстуры, префаба;

- кнопки позволяют прямо в редакторе запустить сцену, поставить на паузу, чтобы проверить состояние каких-нибудь объектов и провести тонкую настройку и контроль путем покадрового выполнения;

- в главном окне редактирования будет проходить вся работа с объектами.

В Unity3D есть собственный подход к написанию шейдеров – Surface Shader. Это просто более высокий уровень абстракции. При компиляции Surface шейдера компилятор создаст шейдер, состоящий из вершинного и пиксельного. Язык в Unity3D свой, называется [ShaderLab](http://docs.unity3d.com/Documentation/Components/SL-Reference.html). Он поддерживает вставки на CG и HLSL.

Первым ключевым словом идет **Shader**. После него в кавычках указывается имя шейдера. После этого идет описание параметров **Properties { }**, которые будут видны в инспекторе и с которыми могут взаимодействовать другие подпрограммы(функции).   
 Каждый шейдер в Unity3D содержит в своем теле как минимум один **Subshader**, который содержит код, используемый программой.

Для описания нашей модели потребуются следующие параметры:

\_MainTex и \_DissolveMap – текстуры;

\_DissolveVal и \_LineWidth – ползунки степени эффекта отражений;  
переменная Color – позволит определить цвет края эффекта.

Программа имеет следующую входную структуру:

uv\_MainTex  и  uv\_DissolveMap – это UV координаты, необходимые шейдеру для правильного наложения текстур на объект.

Выходная структура:

Переменная **о**.

При написании кода мы будем использовать вставку на языке CG. Вставка (директива) обрамляется двумя ключевыми словами: **CGPROGRAM**, **ENDCG**.

**#pragma surface surf Lambert** – объявление функции surface-шейдера и дополнительные параметры. В данном случае функция называется **surf**, а в качестве дополнительных параметров указана модель освещения по Ламберту (простейшая модель освещения — чисто диффузное освещение. Считается, что свет, падающий в точку, одинакового рассеивается по всем направлением полупространства).

  
Рис. 5 Код программы на языке шейдеров

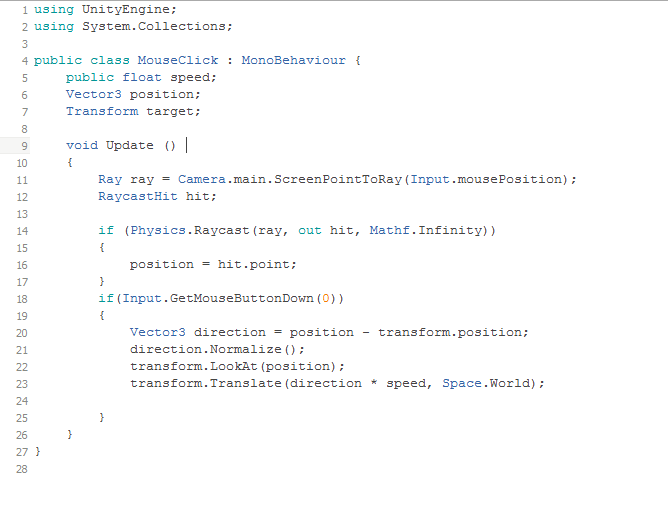


Рис. 6 Код программы на языке С#

**4.1. Результаты экспериментальных запусков**

В двумерном варианте модель рассматривалась на прямоугольной матрице, это вполне удовлетворяло требованиям постановки задачи. В ходе решения задачи в трехмерном случае за модель правого предсердия была выбрана сфера, как наиболее подходящее для визуализации качественного поведения тело. Программа имеет следующий вид:

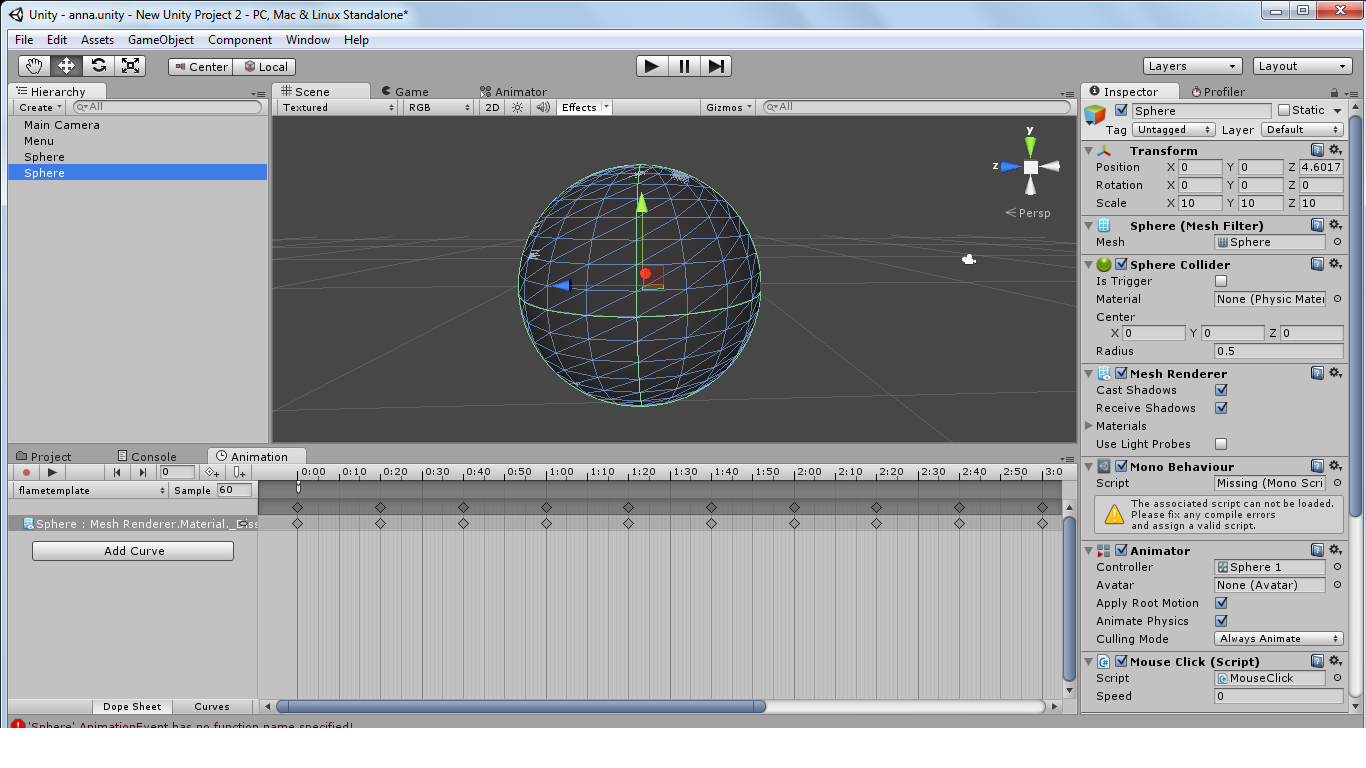


Рис. 7 Интерфейс готовой программы

Интерфейс программы позволяет пользователю:

- выбирать текстуру изучаемой модели (рис.8);

- управлять эффектом вручную с помощью ползунка Dissolve Value (рис.9);

- управлять шириной линии края эффекта (рис.10). Именно от этой ширины линии фронта распространения сигнала зависит время релаксации клетки миокарда;

- выбирать цвет края эффекта (рис.11).

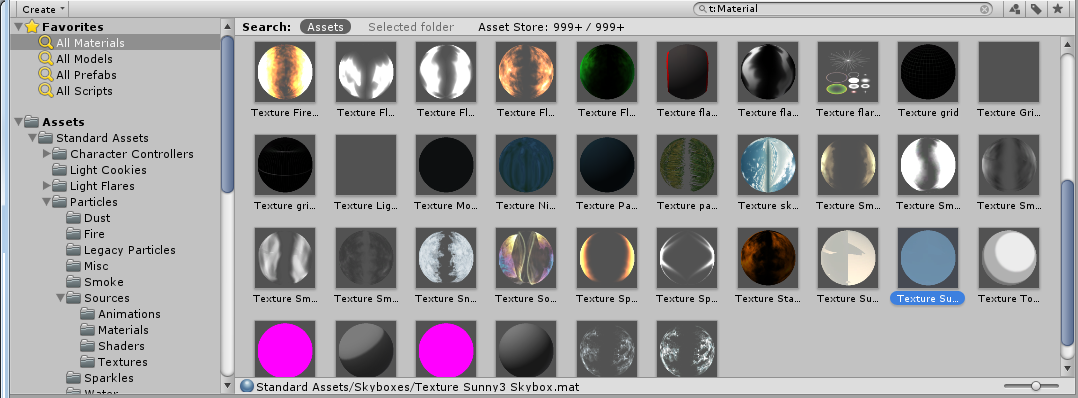


Рис. 8 Выбор текстуры как карты распространения сигнала

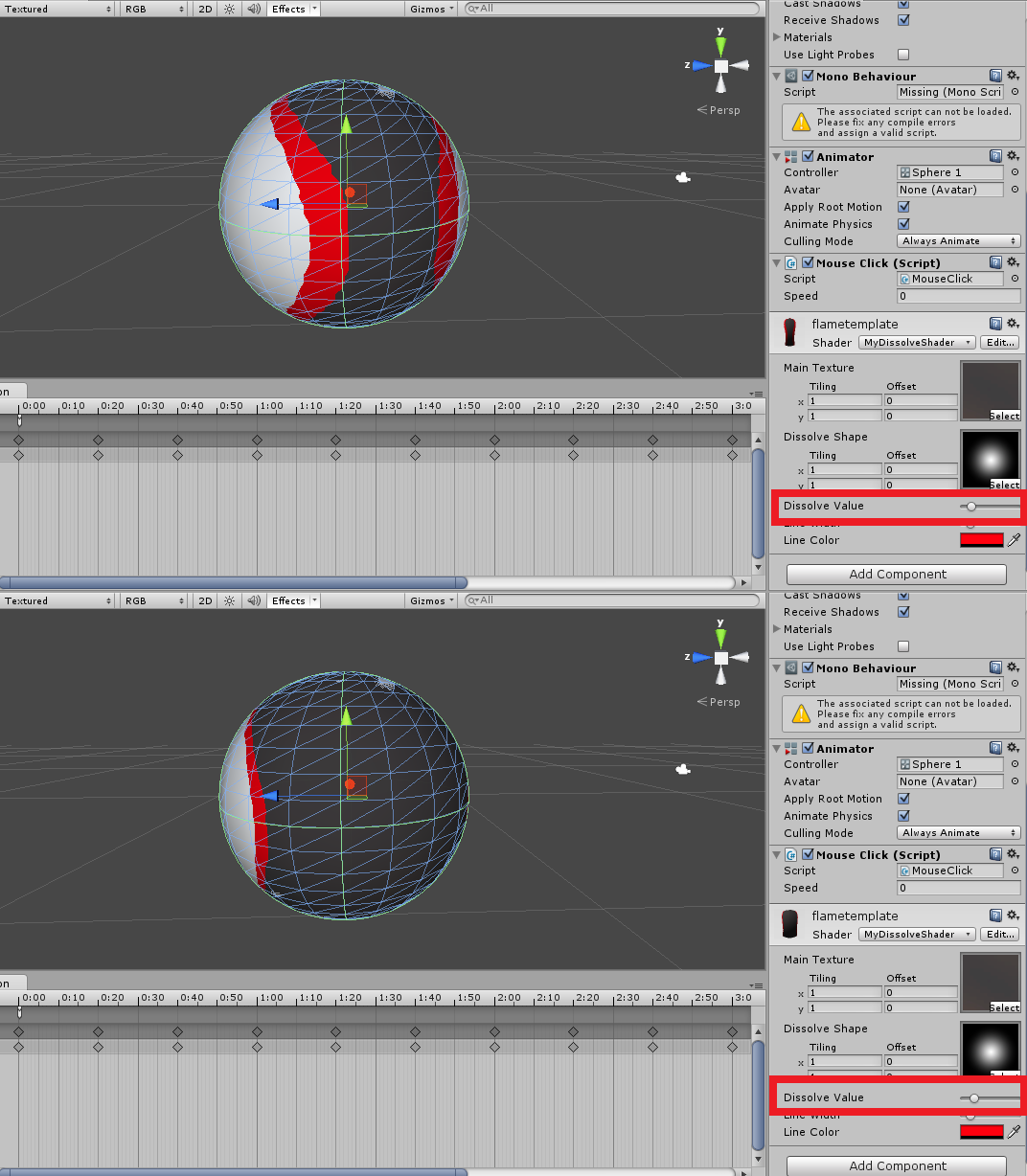


Рис. 9 Управление эффектом вручную

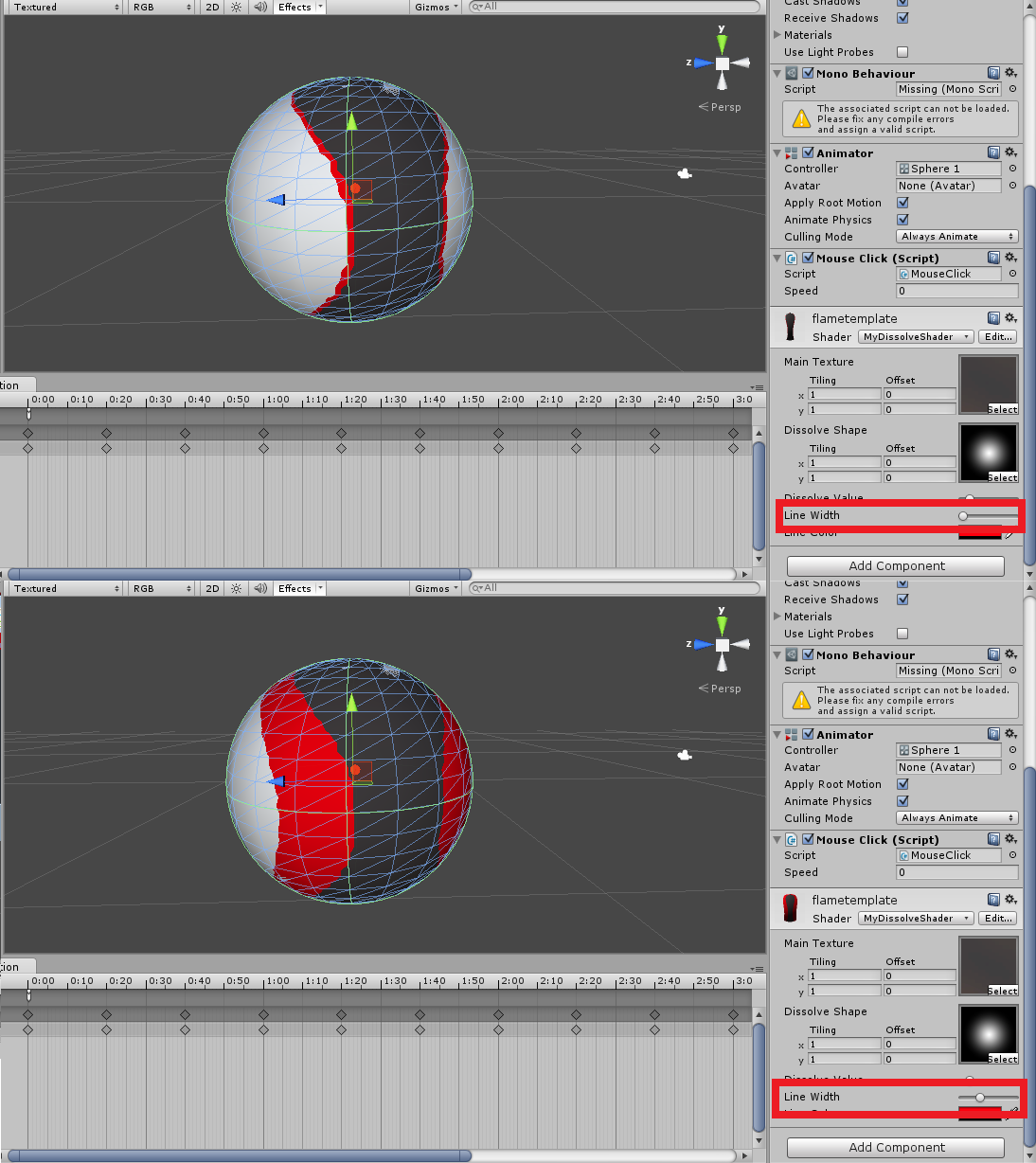


Рис. 10 Управление шириной линии фронта распространения сигнала

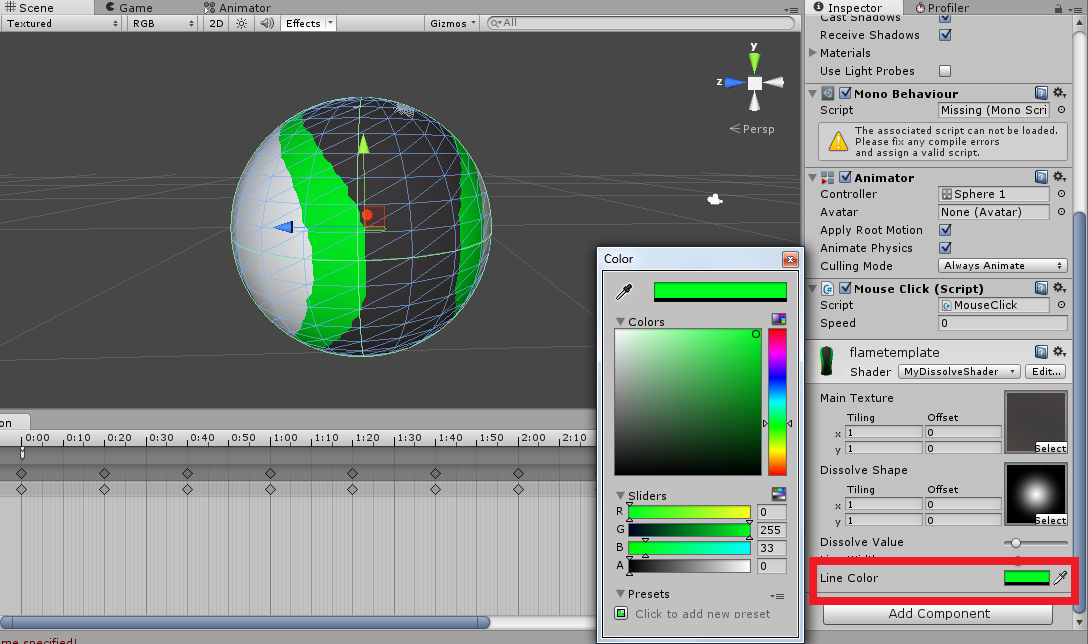


Рис. 11 Выбор цвета края эффекта

Запуск сцены:

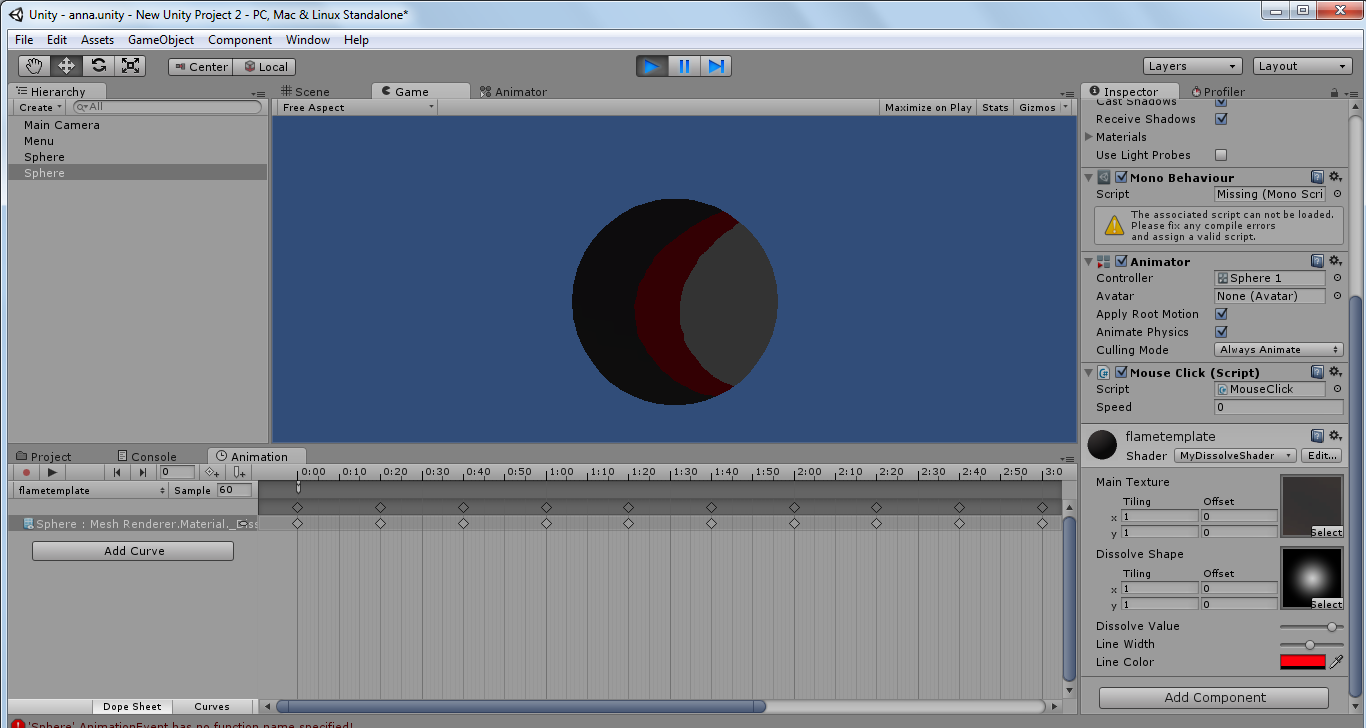
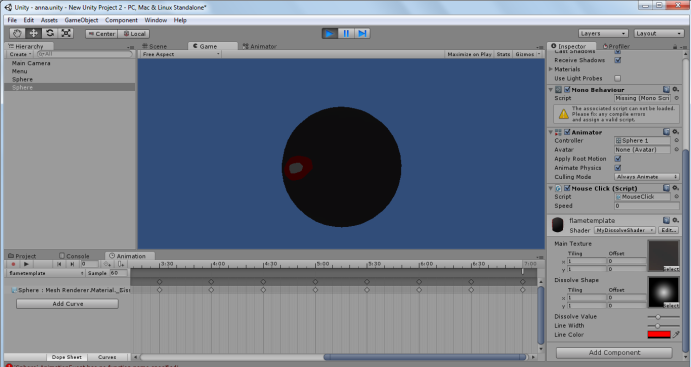
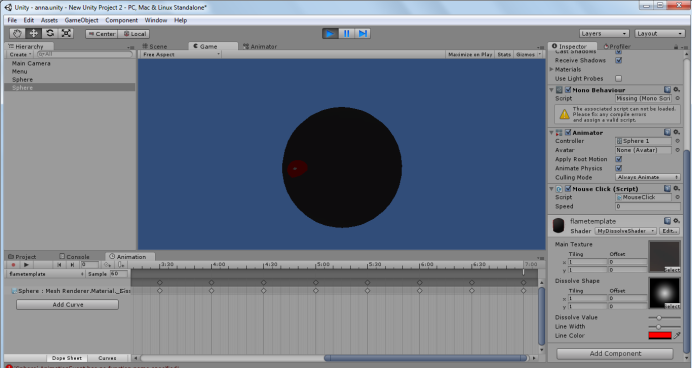
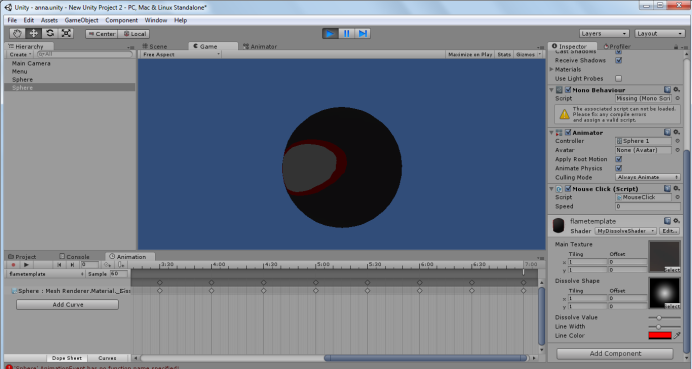
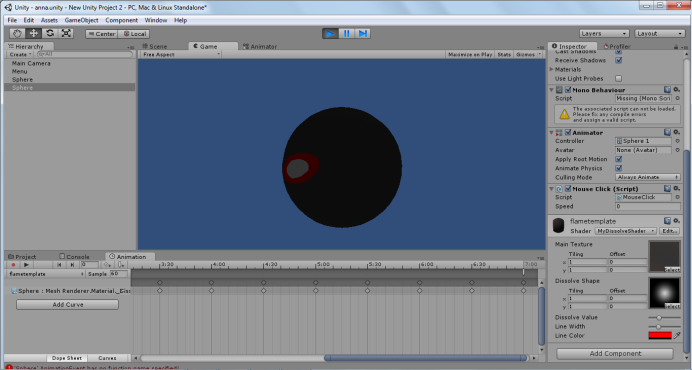
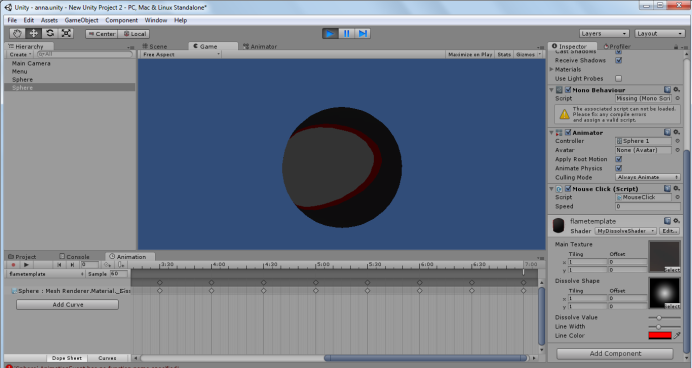
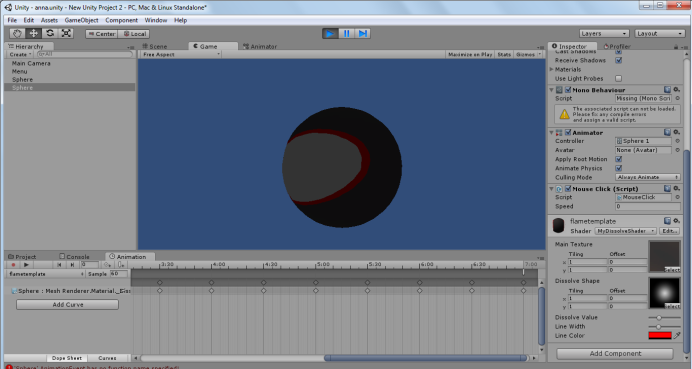


Рис. 12 Программа в действии

При запуске программы хорошо видно имитацию распространения сигнала по поверхности:







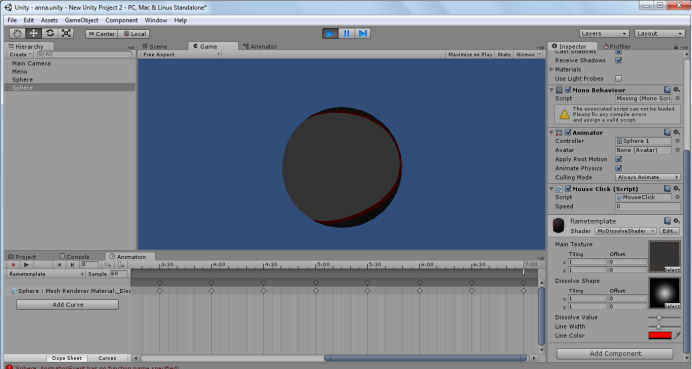
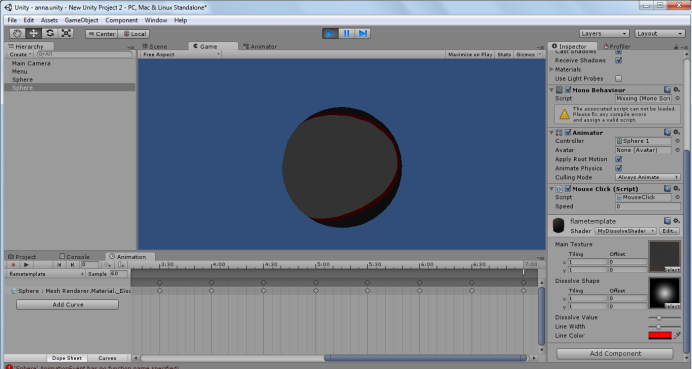


Рис. 13 Распространение сигнала

В данном случае:

светло- серый цвет - клетки в рефрактерном состоянии, то есть невозбудимые к приходящим сигналам;

темно-красный цвет - клетки в возбужденном состоянии, то есть активные клетки;

темно- серый цвет - клетки в невозбужденном состоянии.

**ЗАКЛЮЧЕНИЕ**

В данной работе разработана визуальная качественная модель прохождения сигнала по ткани миокарда. В отличие от физиологической модели, здесь большое внимание уделено именно визуализационной составляющей.

Модель реализована на базе графических процессоров. Использование шейдеров позволило создать мульти-поточное приложение в режиме реального времени. Система может быть использована специалистами при моделировании как нормальных, так и патологических работах сердца, а также может быть применима в учебных целях.

В перспективе возможно усложнение модели путем добавления многослойности, усложнение описания единичной клетки, добавление возможности изменения геометрии тела в реальном времени.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Боресков А. В. Расширения OpenGL. — БХВ-Петербург, 2005.
2. Боресков А. В. Разработка и отладка шейдеров. — БХВ-Петербург, 2006.
3. «Orange Book» — OpenGL Shading Language by Randi J. Rost, Bill M. Licea-Kane, Dan Ginsburg and John M. Kessenich.
4. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн, Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296.
5. Ананий В. Левитин Глава 5. Метод уменьшения размера задачи: Поиск в глубину // Алгоритмы: введение в разработку и анализ = Introduction to The Design and Analysis of Aigorithms. — М.: «Вильямс», 2006. — С. 212-215.
6. Bocharov G.A. Modelling the dynamics of LCMV infection in mice: conventional and exhaustive CTV responses. J. Theor. Biol. 1998. V. 192. P. 283-304.
7. Тоболин Д. Ю., Авербух В. Л., Авербух Н. В., Наймушина А. В., Семенищев Д. В. Деятельный подход при проектировании человеко- компьютерного взаимодействия: на примере медицинских интерфейсов, издательство ЛЕНАНД, 2013 год.
8. С.Ф. Правдин, Л.Б. Кацнельсон, О.Э. Соловьева, В.С. Мархасин, А.В. Панфилов. Численное исследование влияния анизотропии миокарда на его электрофизиологические свойства. Соврем. проблемы математики: тез. Междунар. (44-й Всерос.) молодеж. шк.-конф.- ИММ УрО РАН, 2013.- С.128-131.
9. Борзяк Э. И., Бочаров В. Я., Сапин М. Р. и др. Анатомия человека. В 2 томах / Под ред. акад. РАНМ, проф. М. Р. Сапина. — М.: Медицина, 1993. — Т. 2. — 560 с.
10. Logan R.L., Scott P.J. Uncertainty in clinical practice: implications for quality and costs of health care. Lancet 1996; 347: 595-598.
11. Rosenquist M., Vallin H., Edhagb O. Clinical and electro-physiologic course of sinus node disease: Five-year follow-up study. Am. Heart. J. 109 (1985): 513.
12. Andersen H.R., Thuesen L., Bagger J.P., Vesterlund T., Thomsen P.E. Sick sinus syndrome. Pacemaker with atrial versus ventricular stimulation. Ugeskr-Laeger. 158 (1996); 8: 1062-7.
13. Анатомия и физиология проводящей системы сердца // Клиническая аритмология / Под ред. проф. А. В. Ардашева. — М.: МЕДПРАКТИКА-М, 2009. — С. 35—41. — 1220 с.
14. Kuck K-H, Cappato R., Siebels J. ICD Therapy. Future Publishing Company, Armouk, NY, 1996, p.52-56.
15. Zipes D. In: New Release. National Institutes of Health. April 14, 1997.
16. Revishvili A.Sh., Merkely B., Schaldach M. Transvenous systems for single and dual cham-ber ICD`s: actual and future developments. Mon-duzzi Editore (1997), 527-534.
17. Н.Бейли, Математика в биологии и медицине, издательство "МИР", М.: 1970.
18. Кюрегян С., Матевосян П., Гаспарян О., Мнацаканян М., Мхитарян А., Петросян Т. О применении некоторых задач управления в биомедицинских системах. Сб. материалов годичной научной конференции ГИУА. Т. 2. -Ереван, 2004. - С. 458-462.
19. Солодянников Ю. В. Элементы математического моделирования и идентификация системы кровообращения. Самара: Изд-во Самарского университета, 1994. 315 с.
20. Uttamsingh R. J., Leaning M. S., Bushman J. A. et al. Mathematical model of the human renal system Med. Biol. Eng. Comput. 1985. V. 23. P. 525— 535.
21. С.Ф. Правдин, А.В. Панфилов, В.И. Бердышев, Л.Б. Кацнельсон, О.Э. Соловьёва, В.С. Мархасин. Математическая модель миокарда левого желудочка человека: верификация и расчёт электрического возбуждения. Соврем. проблемы математики: тез. Междунар. 43-й Всерос. мол. шк.-конф.- ИММ УрО РАН, 2012.- С.383-385.
22. Шмидт Р., Тевс Г. Физиология человека, т.2. – М.: Мир, 1996. - 313 с.
23. Фолков В., Нил Э. Кровообращение. - М.:Медицина, 1981. - 600 с.
24. Ашметков И.В., Мухин С.И., Соснин Н.В., Фаворский А.П., Хруленко А.Б. Анализ и сравнение некоторых аналитических и численных решений задач гемодинамики. Дифференциальные уравнения. – 2000. Т.36. №7. - с.919-924.

**ПРИЛОЖЕНИЕ**

Код программы на языке шейдеров.

Shader "MyDissolveShader"

{

Properties

{

\_MainTex("Main Texture", 2D) = "white"{}

\_DissolveMap("Dissolve Shape", 2D) = "white"{}

\_DissolveVal("Dissolve Value", Range(-0.2, 1.2)) = 1.2

\_LineWidth("Line Width", Range(0.0, 0.2)) = 0.1

\_LineColor("Line Color", Color) = (1.0, 1.0, 1.0, 1.0)

}

SubShader

{

Tags{ "Queue" = "Transparent"}

Blend SrcAlpha OneMinusSrcAlpha

CGPROGRAM

#pragma surface surf Lambert

sampler2D \_MainTex;

sampler2D \_DissolveMap;

float4 \_LineColor;

float \_DissolveVal;

float \_LineWidth;

struct Input

{

half2 uv\_MainTex;

half2 uv\_DissolveMap;

};

void surf (Input IN, inout SurfaceOutput o)

{

o.Albedo = tex2D(\_MainTex, IN.uv\_MainTex).rgb;

half4 dissolve = tex2D(\_DissolveMap, IN.uv\_DissolveMap);

half4 clear = half4(0.0);

int isClear = int(dissolve.r - (\_DissolveVal + \_LineWidth) + 0.99);

int isAtLeastLine = int(dissolve.r - (\_DissolveVal) + 0.99);

half4 altCol = lerp(\_LineColor, clear, isClear);

o.Albedo = lerp(o.Albedo, altCol, isAtLeastLine);

o.Alpha = lerp(1.0, 0.0, isClear);

}

ENDCG

}

}

Код программы на языке С#.

using UnityEngine;

using System.Collections;

public class MouseClick : MonoBehaviour {

public float speed;

Vector3 position;

Transform target;

void Update ()

{

Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

RaycastHit hit;

if (Physics.Raycast(ray, out hit, Mathf.Infinity))

{

position = hit.point;

}

if(Input.GetMouseButtonDown(0))

{

Vector3 direction = position - transform.position;

direction.Normalize();

transform.LookAt(position);

transform.Translate(direction \* speed, Space.World);

}

}

}