Министерство образования Республики Беларусь

Учреждение образования

«БелорусскиЙ государственный университет

информатики и радиоэлектроники»

Факультет непрерывного и инновационного обучения

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Основы программирования информационных систем»

|  |
| --- |
| *К защите допустить*: |
| Руководитель курсовой работы  ассистент |
| \_\_\_\_\_\_\_\_\_\_\_\_\_Пархоменко К.А.  \_\_\_.\_\_\_. 20\_\_\_ |

**Пояснительная записка**

к курсовой работе

на тему

**Проектирование графического пользовательского интерфейса для генерации аудио потока с функцио-нально заданной спектрограммой**

Вариант 15

|  |  |  |
| --- | --- | --- |
| Студент | (подпись студента) | Старовойтов Е.А. |
|  |  | Курсовая работа представлена на проверку \_\_\_.\_\_\_. 2017 |
|  |  | (подпись студента) |

Минск 2017

**Содержание**

1. Введение…………………………………………………………………………...... 2
2. Математическое описание и алгоритмы генерирования векторных

Аудиофайлов……………………………………………………………………….. 6

1. Объектно-ориентированные технологии программирования………………….. 17
2. Проектирование графического пользовательского интерфейса

средствами COM-объектов………………………………………………………. 22

1. Алгоритм функционирования программного обеспечения…………………….. 24
2. Web Audi API для генерации звука………………………………………………. 25
3. Анализ результатов решения поставленных задач……………………………… 27
4. Заключение………………………………………………………………………… 28
5. Список использованных источников…………………………………………….. 29
6. Приложение А(Листинг кода)…………………………………………………… 30
7. Приложение Б(Блок-схема алгоритмов)…………………………………………..

**Введение**

Актуальность темы курсовой работы обусловлено широким спектром разнообразных звуков которые нас окружают. С самого рождения человек наделен слухом и это дает ему возможность ощутить весь богатый мир звуков которые могут создавать все объекты в нашем окружении. Но с развитием новых технологий у нас появилась возможность генерировать их программно с различной степенью сложности и частоты. Даже современная музыка во многом основана на генерируемых звуков в смеси с естественными звуками, но так же в обработке средствами программного обеспечения.

Учитывая развития технологий в области обработки звука и современных языков программирования специалисты в этих областях высоко востребованы. Сфера развлечения все больше места занимает в нашей жизни и генерация новых, неизведанных аудио потоков очень привлекательна для каждого. Программное обеспечение которое используют мировые звезды электронной музыки по больешй части основано на цифровой обработке или генерации звуков и сэмплов (Семпл, сэмпл (англ. sample [ˈsɑ:mpəl]) — относительно небольшой оцифрованный звуковой фрагмент. В качестве семпла чаще выступает звук акустического инструмента (например, рояля Steinway, литавр, флейты и т.п.), но также и звуки электромузыкальных инструментов (например, Родес-пиано). Если рассмотреть количество людей увлекающихся электронной музыкой то нельзя не отметить актуальность такого подхода в обработке и создания звуков для современной молодежи.

С технической точки зрения такой кусочек звука(сэмпла) можно представить как N-байтовый элемент массива, в текущем представлении данных. Аналоговый сигнал, предварительно дискретизированный в цифровой формат, может представлять собой элемент: изображения в компьютерной графике (RGB), радиолокационное изображение (GEOTIFF), синтезируемая волна (WAV).

Важны две характеристики семплов: разрешение, то есть число битов, и частота дискретизации (семплирования). Количество битов семпла определяет его способность различать уровни амплитуды сигнала, большое число битов позволяет воспроизводить оцифрованный звук более качественно и точно. Частота семплирования напрямую связана с наивысшей частотой, которая может быть воспроизведена семплом. Например, семпл с частотой дискретизации 40 кГц может воспроизвести звук с частотами до 20 кГц. Это означает, что низкий коэффициент семплирования может быть использован для оцифровки басов, низкочастотных звуков. В то время как для воспроизведения высокочастотных звуковых сигналов (тарелки, хэт, фортепиано) требуется высокая частота семплирования.

Семплы хорошо используется диджеями для создания клубной, транс-, техно-музыки. Подбирая такие семплы по тональности, ритмичности можно в конечном итоге получить готовую музыку не теряя времени и с высоким качеством звучания.

В конце 1970-х годов был создан цифровой музыкальный инструмент, в котором реализован принципиально иной подход к синтезу музыки, получивший название sampling. В прямом смысле это слово означает отбор образцов. Синтезаторы реализующие такой принцип называются семплерами, а образцы звучания — семплами. В процессе записи семплы оцифровываются и сэмплирируются – разбиваются на готовые «кусочки» звуков.

Метод воспроизведения семплов позволяет добиться высокой реалистичности. Причина заключается в том, что устройства воспроизведения семплов имеют дело с акустическими и синтетическими звуками реальных музыкальных инструментов. Когда устройство воспроизведения семплов получает сообщение Note On, то вместо того, чтобы создавать звук, оно воспроизводит цифровой семпл, который может содержать любой реальный звук — от скрипки до детского плача, звук любого синтезатора или драм-машинки.

С такими звуковыми компонентами можно делать всё, что угодно. Можно оставить их такими, как есть, и семплер будет звучать голосами, почти неотличимыми от голосов инструментов-первоисточников. Можно подвергнуть семплы обработке, модуляции, фильтрации, воздействию эффектов и получить самые фантастические, неземные звуки.

В отличии от обычных аудиозаписей, сэмплы имеют незначительный размер и длительность и раскрывают себя в групповой работе. К тому же, фактически существует несколько видов семплов: разовые семплы (One-shot samples), которые обычно используются для создания звуковых эффектов или ударных звуков и воспроизводятся один раз от начала до конца, цикличность отсутствует, и циклические семплы (Loop samples), также называемые семплерными петлями или лупами (Loops), — они имитируют целые инструментальные партии, например, четыре такта партии ударных инструментов.

На данный момент, благодаря широкому распространению программных устройств воспроизведения семплов, появилась возможность создавать полноценные музыкальные произведения путём соединения циклических семплов, при этом ничего более не требуется.

**Цели:**

* Изучить алгоритмы генерации цифровых аудиопотоков
* Реализовать технологии ОО программирования
* Получить навыки для работы в дальнейших проектах

**Задачи:**

* Построить графический и пользовательский интерфейс
* Описать процесс генерации цифровых аудипотоков
* Создать программу для генерации звука
* Изучить возможные способы получения цифрового звука
* Построить блок-схему описывающую работу программы

**2. Математическое описание и алгоритмы генерирования векторных аудиофайлов**

**Звук** - волнообразные колебания твердых, жидких и газообразных тел воспринимаются органами слуха - ушами, в ввиде особых ощущений, звуков. Звуковые волны распространяются от получившего удар и приведенного в дрожательное колебание тела во все стороны окружающей тела среды. Все тела по отношению к звуку делятся на проводящие и на неупругие (например - воск) и потому не проводящие звук. В качестве проводника звука чаще всего выступает воздух. В зависимости от среды в коротой звук распростроняется он может иметь различную скорость, в воздухе при 0° в 1 сек.332,5 м. В дождливую погоду при громе можно заметь скорость его распространения, в детстве многие отсчитывали секунды после раската грома и таким образом высчитывали дальность от наблюдателя. Гром, несмотря на то, что возникает одновременно с разрядом электричества, молнией, слышен через некоторый промежуток времени, в зависимости от отдаленности места электрического разряда. Звук характеризуется высотой, силой и оттенком. Высота звука зависит от числа колебаний звучащего тела; сила звука - в данном месте есть количество звуковой энергии, проходящей в единицу времени через единицу площади, перпендикулярной к направлению распространения звука. Оттенок звука или тембр зависит от высших или гармонических тонов, сопровождающих во всяком звуке основной тон.

Энергетической характеристикой звуковых колебаний является интенсивность звука - энергия, переносимая звуковой волной через единицу поверхности, перпендикулярную направлению распространения волны, в единицу времени. Интенсивность звука зависит от амплитуды звукового давления, а также от свойств самой среды и от формы волны. Субъективной характеристикой звука, связанной с его интенсивностью, является громкость звука, зависящая от частоты. Наибольшей чувствительностью человеческое ухо обладает в области частот 1-5 кГц.

Источниками звука могут стать любые явления, вызывающие местное изменение давления или механическое напряжение. Широко распространены источники звука в виде колеблющихся твёрдых тел (например, диффузоры громкоговорителей и мембраны телефонов, струны и деки музыкальных инструментов; в ультразвуковом диапазоне частот - пластинки и стержни из пьезоэлектрических материалов или магнитострикционных материалов). Источниками звука могут служить и колебания ограниченных объёмов самой среды (например, в органных трубах, духовых музыкальных инструментах, свистках и т.п.). Сложной колебательной системой является голосовой аппарат человека и животных. Возбуждение колебаний источников звука может производиться ударом или щипком (колокола, струны); в них может поддерживаться режим автоколебаний за счёт, например, потока воздуха (духовые инструменты). Обширный класс источников звука - электроакустические преобразователи, в которых механические колебания создаются путём преобразования колебаний электрического тока той же частоты. В природе звук возбуждается при обтекании твёрдых тел потоком воздуха за счёт образования и отрыва вихрей, например при обдувании ветром проводов, труб, гребней морских волн. Звук низких и инфранизких - частот возникает при взрывах, обвалах. Многообразны источники акустических шумов, к которым относятся применяемые в технике машины и механизмы, газовые и водяные струи.

Приёмники звука - служат для восприятия звуковой энергии и преобразования её в другие формы. К приёмникам звука относится, в частности, слуховой аппарат человека и животных. В технике для приёма звука применяются главным образом электроакустические преобразователи: в воздухе - микрофоны, в воде - гидрофоны и в земной коре - геофоны.

Чистый звуковой тон представляет собой звуковую волну, подчиняющуюся синусоидальному закону:

у =am\* sin (wt) = аm\*sin (2пft),

где :

am - максимальная амплитуда синусоиды;

w - частота (w=2пf);

f - количество колебаний упругой среды в секунду (f=1T);

Т-период;

t - время (параметрическая переменная).

Звук характеризуется частотой (f), обычно измеряемой в герцах, т.е. количеством колебаний в секунду, и амплитудой (у). Амплитуда звуковых колебаний определяет громкость звука. Для монотонного звука (меандр.) характерно постоянство амплитуды во времени. Затухающие звуковые колебания характеризуются уменьшением амплитуды с течением времени.

Человек воспринимает механические колебания частотой 20 Гц - 20 КГц (дети - до 30 КГц) как звуковые. Колебания с частотой менее 20 Гц называются инфразвуком, колебания с частотой более 20 КГц - ультразвуком. Для передачи разборчивой речи достаточен диапазон частот от 300 до 3000 Гц.

Если несколько чистых синусоидальных колебаний смешать, то вид колебания изменится - колебания станут несинусоидальными. Особый случай, когда смешиваются не любые синусоидальные колебания, а строго определенные, частота которых отличается в два раза (гармоники). Основная гармоника имеет частоту, и амплитуду а1; вторая гармоника - частоту f2 и амплитуду а2; третья гармоника соответственно f3 и a3. Причем f1<f2<f3, а1>а2>а3,При бесконечном количестве таких гармоник образуется периодический сигнал, состоящий из прямоугольных импульсов. На слух всякое отклонение от синусоиды приводит к изменению звучания. В IBM PC источником звуковых колебаний является динамик (PC Speaker), воспроизводящий частоты приблизительно от 2 до 8 КГц. Для генерации звука в PC Speaker используются прямоугольные импульсы. Синусоидальные сигналы в ЭВМ можно получить только с помощью специальных устройств - аудиоплат.

Большие объёмы аналоговых звуковых сигналов нуждаются в цифровой обработке для дальнейшего их использования. Для этого могут применятся различные способы и методы. Самый распространённый способ является кодовая модуляция – импульсно-кодовая ( PCM ) и плотностно-импульсная ( PDM) модуляции. Почти все аналоговые звуковые сигналы допускают такие методы преобразования.

**PCM**

Первый шаг для преобразования сигнала из аналого-цифрового должен отфильтровать компонент с более высокой частотой сигнала. Это делает вещи более легким нисходящим для преобразования этого сигнала. Большая часть энергии разговорного языка где-нибудь между 200 или 300 герц и приблизительно 2700 или 2800 герц. Примерно и стандартная голосовая связь с 3000 ширинами полосы пропускания (в Гц) для стандартного речевого сигнала установлена. Поэтому у них не должно быть точных фильтров (это очень дорого). Пропускная способность 4000 герц сделана из точки оборудования если представление. Этот фильтр, ограничивающий полосу, используется для сглаживания. Такое случается, если входящий аналоговый голосовой сигнал оцифрован с недостаточной дискретизацией, что по критерию Нюквиста как Fs < 2(BW). Частота дискретизации является меньше, чем максимальная частота входного аналогового сигнала. Это создает наложение между спектром частот выборок и входным аналоговым сигналом. Фильтр исходящих данные низких частот, используемый для восстановления исходного входящего сигнала, не достаточно умен для обнаружения этого наложения. Поэтому это создает новый сигнал, который не происходит из источника. Это создание ложного сигнала, когда выборку вызывают, искажая.

При импульсно-кодовой модуляции осуществляется преобразование передаваемого аналогового сигнала в его цифровую форму путем применения к нему следующих операций:

* Дискретизация по времени
* Квантования по амплитуде
* Кодирование

Для осуществления такого преобразования аналогового сигнала в цифровой применяется устройство АЦП (аналого-цифровой преобразователь). АЦП – это электронное устройство которое преобразует передаваемое напряжение в двоичный цифровой код. И хотя они могут иметь различные типы, в целом все они выполняют одну и туже задачу.

Аналого-цифровой преобразователь через равные промежутки времени измеряет амплитуду передаваемого аналогового сигнала, в промежутке получает обрабатывает ее и получает новые данные где на выходе преобразует эти данные в двоичный цифровой код.

PCM имеют несколько разновидностей:

* Дифференциальная – сигнал кодируется в виде разности между текущими и предыдущим измеренным значением.
* Адаптивная – изменение шага измерения, что позволяет уменьшить требования к полосе пропускания.
* Линейная

В качестве практического применения импульсно-кодовой модуляции можно выделить преобразования в голосового аудио сигнала в мобильной и IP-телефонии и преобразования аналоговых аудио сигналов в цифровой вид для хранения и дальнейшего использования. Из популярных форматов хранения цифровых записей существуют: MP3, FLAC, APE, OGG, WAV, WMA и др.

**PDM**

Плотностно-импульсная модуляция или PDM - это форма модуляции, используемая для представления аналогового сигнала с двоичным сигналом. В сигнале PDM определенные значения амплитуды не кодируются в кодовые слова импульсов разного значения, поскольку в таком случае они были бы в импульсно-кодовой модуляции (PCM).

Большую популярность приобрела разновидность PDM, такая как Широтно-импульсная модуляция (PWM) и она является частным случаем PDM, где частота переключения фиксирована, и все импульсы, соответствующие одному образцу, смежные в цифровом сигнале.

Для 50% -ного напряжения с разрешением 8 бит, сигнал PWM включается на 128 тактовый циклов , а затем выключается для остальных 128 циклов. С PDM и той же тактовой частотой сигнал будет чередоваться между каждым циклом и выключением. Среднее значение составляет 50% для обеих осциллограмм, но сигнал PDM переключается чаще. Для уровня 100% или 0% они одинаковы.

В битовом потоке модуляции импульсной плотности 1 соответствует импульсу положительной полярности (+ A), а 0 соответствует импульсу отрицательной полярности (-A). Математически это можно представить как:

**x[n] = -A(-1)a[n]**

где:

x[n] – биполярный поток битов (или –A или +A)

a[n] – соответствующий бинарный поток битов (либо 0, либо 1)

Запуск состоящий из всех 1, будет соответствовать максимальному(положительному) значению амплитуды, все 0 будут соответствовать минимальному(отрицательному) значению амплитуды, а переменные значения 1 и 0 будут соответствовать нулевой амплитуде.

Форма непрерывной амплитуды восстанавливается посредством низкочастотной фильтрации биполярного битового потока плотностно импульстной модуляции.

Пример работы:

Один период тригонометрической синусоиды отобранной 100 раз и представленный как поток бит PDM, представляет собой:

0101011011110111111111111111111111011111101101101010100100100000010000000000000000000001000010010101



Два периода синусоидальной волны с более высокой частотой будут выглядеть так:

0101101111111111111101101010010000000000000100010011011101111111111111011010100100000000000000100101



При плотностно импульсной модуляции на пике синусоидальной волны наблюдается высокая плотность 1, а на падении волны – малая плотность 1.

PDM используется как для преобразования с аналогового в цифровой сигнал так и возможет обратный процесс.

**Аналоговой в цифровой**

При преобразовании из аналогового в цифровой битовой поток PDM преобразуется в процессе сигма-дельта модуляции, который обеспечивает оцифровку сигнала с заданными характеристиками в рабочей полосе частот.

В процессе используется однобитовый квантователь , который выдает на выходе либо 1, либо 0 в зависимости от амплитуды аналогового сигнала. 1 и 0 соответствует сигналу, который находиться на вершине волны или в падении соответственно.

Так как в реальном мире аналоговые сигналы редко распространяются в одном направлении, то допускается ошибка квантования, разница между 1 и 0 и фактической амплитудой, которую она представляет. Эта ошибка отрицательно сказывается в процессе сигма-дельта модуляции. Таким образом, каждая ошибка последовательно влияет на каждое другое измерение квантования и его погрешность. В итоге это приводит к усреднению ошибки модуляции.

**Цифровой в аналоговый**

Процесс декодирования PDM сигнала в аналоговой немного проще по своей сложности. Для этого нужно передавать PDM сигнал через фильтр нижних частот. По существу функция фильтра нижних частот усредняет сигнал. Средняя амплитуда импульсов измеряется по плотности этих импульсов с течением времени, поэтому фильтр нижних частот является единственным шагом, который требуется в процессе декодирования цифрового сигнала в аналоговый.

**Аудиопоток**

Аудиопоток – это цифровой или аналоговый звук, который пользователь непрерывно получает от стороннего источника. Первые источники потоковых звуковых данных были радио и теле вещания. В наши дни большой популярностью пользуется онлайн-радио и потоковое видео(стримы).

Первые попытки передачи потоковой информации наблюдались уже в середине ХХ века. Однако виду низкий технических возможностей не удавалось добиться высокого и стабильного качества. С развитием персональных компьютеров ситуация в этом направлении сдвинулась с мертвой точки, но по прежнему слабые процессоры и шины не позволяли получить достаточные результаты.

После 1990 годов пользователи интернета получили в свое распоряжение такие технические достижения как:

* Высокая передача данных по сети
* Увеличилось количество уникальных пользователей
* Стандартизировались форматы и протоколы передачи данных
* Получила развития интернет коммерция
* Повсеместная прокладка сетей

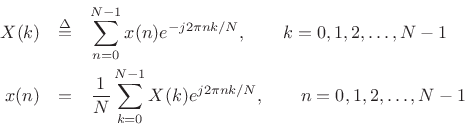
Все это дало доступ к потоковому потоковому аудио и другим мультимедиа данным для широкого круга потребителей. Автономные приемники потоковых аудио и видео данных, такие как домашние мультимедийные центры и приемники онлайн радио.

**Спектр цифрового сигнала**

В целом это понятие можно описать как разложение сигнала на более простые функции(изначальный сигнал описывается в качестве линейной комбинации простых функций). Это похоже на разложение пучка света на составляющие элементы – спектры. Если конечные простые функции представляются как cos/sin, то такое разложение называют «преобразованием Фурье». Если описываемый сигнал цифровой, то чаще всего используется Дискретное преобразование Фурье (ДПФ). Процесс, обратный этому, когда по спектру восстанавливается исходный сигнал, называется Обратным преобразованием Фурье.

Такое разложение сигнала на спектры находит свое применение в анализе прохождения сигнала через электрические цепи. Преимуществом таких разложений в электроцепи изначальных сигналов являются: проходящий по цепи сигнал подвергаются изменению и преобразованию. Элементы цепи, такие как токи и напряжение, под действием сигнала описываются как дифференциальное уравнение, соответствующие этим элементам и их способу взаимодействия. Это позволяет при ожидаемой реакции системы на какой либо входной сигнал, например на синусоидальные колебания определенной частоты, получить реакцию системы на более сложный сигнал, разложив его в ряд на более простые по синусоидальным колебаниям.

Часто применяемое дискретное преобразование Фурье можно представить в качестве формулы:



где:

x(n) - отсчеты цифрового сигнала.

X(k) – спектр цифрового сигнала, комплексные значения.

N– длина исследуемого окна анализа, чем больше исследуемое окно, тем точнее получаемые частотные компоненты и тем больше времени необходимо для полного анализа.

**Спектрограмма**

Спектрограмма – это графическое изображение спектра сигнала, выполненное анализатором спектра. Чаще всего представляет собой графический компьютерный файл но может и иметь другой вид. Такое представление сигнала находит применение при решении самый разных задач – от обработки естественной человеческой речи и звуков до радио и гидролокации.

Спектрограмма может быть представлена в разных представлениях – двумерное представление, где в качестве вертикальных и горизонтальных осей выступают время и частота, или трехмерная модель представления, где к предыдущим осям может добавится измерение амплитуды сигнала на определённой частоте.

По способам формирования спектрограммы выделяют следующие варианты:

* Метод полосовых фильтров – входные сигналы аппроксимируется в качестве наборов фильтров. Устаревший способ по преобразованию входного аналогового сигнала на частотные диапазоны.
* Метод оконного преобразования Фурье – благодаря цифровой обработке, преобразование осуществляется по временному сигналу. Входной сигнал разбивается на отдельные блоки, затем каждая часть перекрывается и преобразуется по законам Фурье, для расчета значения для каждого рассчитываемого участка. Каждый полученная часть представляет собой в конечном итоге вертикальную линию на изображении.

Математически такое преобразование имеет вид:

spectrogram (t, w) = |STFT (t, w) |2

где:

t – время исследуемого промежутка

w – ширина

STFT – кратковременное преобразование Фурье

Спектрограмма часто позволяет найти в записи дефекты, которые неочевидны при прослушивании, но могут сказаться при последующей обработке. Спектр и спектрограмма – способы представления звука, более близкие к слуховому восприятию, нежели осциллограмма.

**Понятие изображения и классификация цветовых моделей**

Изображение - это двумерное графическое представление какого-либо объекта. Мы рассмотрим цифровые изображения которые в зависимости от описываемого способа могут делиться на растровые и векторные.

* Растровые – изображения которые представляет собой сетку пикселей – цветных точек, которые в зависимости от разрешения изображения( количества пикселей на площадь изображения) могу по разному восприниматься человеческим глазом по качеству.
* Векторные – изображения, которые представляют собой совокупность математические описанных геометрических фигур. Идеально подходит для описания простых фигур без большого количества деталей.

Цветовая модель – абстрактная модель представления цветов в виде кортежей чисел, обычно из трёх или четырёх значений, называемых цветовыми компонентами или цветовыми координатами. Вместе с методом интерпретации этих данных множество цветов цветовой модели определяет цветовое пространство. Цветовая модель задаёт соответствие между воспринимаемыми человеком цветами, хранимыми в памяти, и цветами, формируемыми на устройствах вывода (возможно, при заданных условиях).

Применительно к компьютерной графике описание цвета также должно учитывать специфику аппаратуры для ввода/вывода изображений. В связи с необходимостью описания различных физических процессов воспроизведения цвета были разработаны различные цветовые модели. Цветовые модели позволяют с помощью математического аппарата описать определенные цветовые области спектра. Цветовые модели описывают цветовые оттенки с помощью смешивания нескольких основных цветов.

Основные цвета разбиваются на оттенки по яркости (от темного к светлому), и каждой градации яркости присваивается цифровое значение (например, самой темной – 0, самой светлой – 255). Считается, что в среднем человек способен воспринимать около 256 оттенков одного цвета. Таким образом, любой цвет можно разложить на оттенки основных цветов и обозначить его набором цифр – цветовых координат.

Различные цветовые модели могут отличаться по способу описания цвета и рассмотрим некоторые из них:

**RGB** – Одна из самых распространённых моделей, где основная идея состоим в том, что любой цвет можно получить смешиванием трех основных цветов: красного(**R**ed), зеленого(**G**reen), синего(**B**lue). Каждый канал в таком случаем является отдельным параметром, который указывает на количество соответствующей цвета в конечном цветовом представлении. Например: (255, 64, 23) – цвет, в котором явно преобладает красный компонент, включает немного зелёного и очень мало синего.

Разумеется, такая цветовая модель наиболее подходит для более глубокой передачи всего богатства красок окружающей природы. Но он нуждается и в больших затрат памяти, так как глубина цвета тут наибольшая – 3 канала по 8 бит на каждый, что дает в общей сложности 24 бита. Такую цветовую модель описывают как аддитивную( от англ. Add – добавлять), т.к. мы добавляем разное количество цветовых оттенков для получения итогового результата.

Главное, что нужно понимать, это то, что аддитивная цветовая модель предполагает, что вся палитра цветов складывается из светящихся точек. То есть на бумаге, например, невозможно отобразить цвет в цветовой модели RGB, поскольку бумага цвет поглощает, а не светится сама по себе.

**CMYK, CMY**- в отличии от RGB-модели, которая является аддитивной, CMYK модель является субтрактивная(от англ. Subtract – вычитать). Как и модель выше, цвет формируется по главным цветам, первоначальные буквы которых и формируют название модели – **C**yan, **M**agenta, **Y**ellow и **K**ey color(опционально).

В CMY-модели описываются цвета на белом носителе, т. е. краситель, нанесенный на белую бумагу, вычитает часть спектра из падающего белого света. Например, на поверхность бумаги нанесли голубой (Cyan) краситель. Теперь красный свет, падающий на бумагу, полностью поглощается. Таким образом, голубой носитель вычитает красный свет из падающего белого.

Такая модель наиболее точно описывает цвета при выводе изображения на печать, т. е. в полиграфии.

**HSB** – цветовая модель схожая по сути с RGB, но отличается системой координат в представлении цветовой схемы. Каждый цвет, представляющийся таким способом включает 3 характеристики – тон(**H**ue), насыщенность(**S**aturation) и яркость(**B**rightess). Тон в этом случает выступает базовым цветом, насыщенность – процентное соотношение белого цвета добавленного к базовому цвету и яркость – это процент добавленного черного цвета. В итоге HSB – это трехканальная цветовая модель. Из-за несоответствия строгой математической модели, цвета описанные по такой схеме не соответствуют цвету, воспринимаемым человеческим глазом.

Исходя из задач которые выполняете вам могут понадобиться разные схемы и цветовые модели описываемого цвета.

**Анализ алгоритмов генерации векторных изображений**

Как было рассмотрено ранее в основе построения векторных изображений лежит математические формулы, которые служат в описании элементарных геометрических фигур. Поэтому любые объекты векторной графики являются графическими изображениями математических объектов. При преобразовании таких объектов в растровую графику, они будут называться «примитивами».

В ходе использования векторной графики часто возникает необходимость в автоматической генерации базовых элементов криволинейной формы, в основе которых лежат разнообразные математические формулы. Генерация простых объектов осуществляется на основе базовых графических элементов простой формы путем их дублирования и геометрического преобразования. Далее такой объект можно использовать для формирования более сложной симметричной фигуры. Такая фигура является сложной системой из нескольких подструктур, каждая из который построена на базе векторного объекта определённой формы(базового элемента).

Элементы векторных изображений могут быть одним элементом или включать в себя:

* Точки
* Круги и окружности
* Многоугольники
* Сплайны
* Кривые Безье
* Фракталы

Многочлен Безье повсеместно используется для описания криволинейных векторных контуров. В основе этого метода лежит использование пары отрезков касательных, приложенных к сегменту контура его окончания. Размеры и положение касательных в этом случае выступают в качестве формообразователя сегмента. Математическое представление формирование сегмента по способу Безье можно выразить следующим уравнением:

**p(t) = p0(1-t)3+p13t(1-t)2+p23t2(1-t) +p3t3;**

**0≤t≤1;**

где:

p0, p3 - координаты крайних точек.

p1 и p2 – координаты крайних точек отрезков касательных, проведенных из крайних точек сегмента.

Чаще всего элементы и сегменты векторной графики описывающиеся методом Безье выражаются двумя параметрическими функциями:

**x(t) = x0(1-t)3 + x13t(1-t)2 + x23t2(1-t) +x3t3;**

**y(t) = y0(1-t)3 + y13t(1-t)2 + y23t2(1-t) +y3t3;**

При визуализации такой фигуры, находиться решение для каждого описывающего уравнения в цикле, с нахождением горизонтальных и вертикальных координат точек кривой, соединяющие крайние точки сегмента. Значение t принимает значение от 0 до 1 на протяжении цикла вычисления.

Опираясь на это мы видим что формирования двумерного криволинейного сегмента по методу Безье нам необходимо 8 значений коэффициентов, которые соответствуют 4 точкам координат. В любом случаем для создания векторного изображения математическое уравнения для этого принимает вид y = f(x);

**3.Объектно-ориентированные технологии программирования**

**Объектно-ориентированное программирование (ООП)** - это метод программирования, при использовании которого главными элементами программ являются объекты. В языках программирования понятие объекта реализовано как совокупность свойств (структур данных, характерных для данного объекта), методов их обработки (подпрограмм изменения их свойств) и событий, на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта. Объединение данных и свойственных им процедур обработки в одном объекте, называется инкапсуляцией и является одним из важнейших принципов ООП.

Основными фундаментальными понятиями характеризующие ООП являются:

**Класс** – это комплексный тип данных или шаблон на основе которых создаются конкретные программные обьекты. Он сотоит из набора полей (примитивных типов данных) и методов(функций для работы с этими полями). Каждый обьект имеющий структуру этого обьекты называется экземпляром класса.

**Наследование** – свойства класса которые позволяют создавать новые классы на основе уже существующих и дают возможность наследовать методы и свойства класса родителя.

**Инкапсуляция** – свойство обьектов системы, которое дает возможность обьединить данные и методы, работаюшие с ними в классе. Это позволяет использовать один метод для разных обьектов.

**Абстракция** – выделение значимых характеристик обьекта и абстрагирование от незначимых.

**Полиморфизм** - означает, что рожденные объекты обладают информацией о том, какие методы они должны использовать в зависимости от того, в каком месте цепоч-ки (дерева классов) они находятся, иными словами это концепция, реализующая "множество методов в одном интерфейсе".

**Модульность** – такая организация обьектов, которое заключает в себя полное определение их характеристик и методов внутри себя, это дает возможность сводного перемещения и повторного использования кода.

Развитием объектно-орниентированной парадигмы (методологии), стала объектно-событийная парадигма, опирающаяся на понятия объекта и события. Эта парадигма позволяет конструировать, программировать распределенные вычислительные среды, в том числе среды реального времени, [SCADA](http://bourabai.kz/dbt/scada.htm) и пр.

**Событие в объектно-ориентированном программировании** - это сообщение, которое возникает в различных точках исполняемого кода при выполнении определённых условий. События предназначены для того, чтобы иметь возможность предусмотреть реакцию программного обеспечения. Для решения поставленной задачи создаются обработчики событий: как только программа попадает в заданное состояние, происходит событие, посылается сообщение, а обработчик перехватывает это сообщение. В общем случае в обработчик не передаётся ничего, либо передаётся ссылка на объект, инициировавший (породивший) обрабатываемое событие. В особых случаях в обработчик передаются значения некоторых переменных или ссылки на какие-то другие объекты, чтобы обработка данного события могла учесть контекст возникновения события. Самое простое событие - это событие, сообщающее о начале или о завершении некоторой процедуры. Событие, по сути, сообщает об изменении состояния некоторого объекта. Наиболее наглядно события представлены в пользовательском интерфейсе, когда каждое действие пользователя порождает цепочку событий, которые, затем обрабатываются в приложении. В объектно-ориентированном анализе для описания динамического поведения объектов принято использовать модель состояний. Событие - это переход объекта из одного состояния в другое. Взаимодействие объектов также осуществляется при помощи событий: изменение состояния одного объекта приводит к изменению состояния другого объекта, а событие оказывается средством связи между объектами. Событие - это <абстракция инцидента или сигнала в реальном мире, который сообщает нам о перемещении чего- либо в новое состояние>. Далее, выделяются четыре аспекта события:

|  |
| --- |
| * метка - уникальный идентификатор события; * значение - текстовое сообщение о сути произошедшего; * предназначение - модель событий, которая принимает событие; * данные - данные, которые переносятся от одного объекта к другому; |

Первый ряд примеров событий доставляет собственно сам жизненный цикл объекта:

|  |
| --- |
| * создание объекта; * уничтожение объекта; |

Более сложные примеры событий возникают тогда, когда у объекта появляются внутренние состояния, которые описываются соответствующей диаграммой переходов (из одного состояния в другое).

Современными языками объектно-ориентированного программирования являются [*С++*](http://bourabai.kz/C-Builder/cpp/index.htm) и [*Java*](http://bourabai.kz/alg/classification04.htm#java). С середины 90-х годов многие объектно–ориентированные языки реализуются как [системы визуального программирования](http://bourabai.kz/einf/4gl.htm), в которых интерфейсная часть программного продукта создается в диалоговом режиме, практически без написания программных операторов. На данный момент, на многих языках программирования существует возможность следовать парадигме ООП. И хотя в некоторых случаях они отличаются по реализации, основные принципы остаются одинаковыми.

Самыми известными языками придерживающиеся ООП на данный момент являются C++ и Java, однако большую популярность на данный момент завоевывает Java Script, и хотя этот язык начинал свой путь лишь как инструмент для создание динамичности в Web, сейчас этот мощный язык программирования который занимает второе место по популярности после Java.

**Java Script как капитан в плавании WEB технологий**

Ввиду того что реализация курсовой работы осуществляется на JavaScript то необходимо рассмотреть основные моменты и путь развития этого замечательного языка.

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

Вы можете сделать очень многое с JavaScript. Вы можете начать с малого, с простых функций, таких как карусели, галереи изображений, изменяющиеся макеты и отклик на нажатие кнопок. Когда вы станете более опытным в языке, вы сможете создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JS изначально создавался для добавления динамичности web-страниц. Программы написанные на это языке принято называть *скриптами.* В браузере они напрямую подключаться к с HTML и зависимости от условий выполняются. В общем плане такие скрипты это обычный текст и в этом плане они немного отличаются от программ написанных на таких типизированных языках как Java. Java Script может выполняться не только в браузере но и в любом устройстве где имеется интерпретатор. Из-за этого язык получил широкое распространение в написании программ к устройствам IoT(Internet of Things – с англ. Интернет вещей).

**Что возможно реализовать используя Java Script?**

Современные возможности JS предоставляют разработчикам широкие возможности в реализации своих идей. Это «безопасный» зык программирования общего назначения. Так как изначально язык был ориентирован на браузер, это накладывает определенные ограничения при работе с памятью или процессорами. Что касается остального то оно во многом зависит от окружения в котором используется Java Script. Браузере это широкие возможности по манипуляциям со страницей, взаимодействия с пользователем и частично с сервером:

* Создание новых html-тегов и изменение, удаление существующих, изменение стилей и элементов.
* Взаимодействия и реакция на действия с пользователем – обработка кликов, форм, нажатий клавиш.
* Запрашивать данные, выводить сообщения.
* Отправка запросов на сервер, получения и обработка результат.

Это лишь малая часть того что умеет JS. И хотя если вы используете язык на стороне сервера то у вас, можно сказать, «развязаны руки» рассмотрим ограничения накладываемые браузером на JS.

**Что не умеет Java Script?**

JavaScript – быстрый и мощный язык, но браузер накладывает на его исполнение некоторые ограничени. Это сделано для безопасности пользователей, чтобы злоумышленник не мог с помощью JavaScript получить личные данные или как-то навредить компьютеру пользователя.

Кроме того, современные браузеры предоставляют свои механизмы по установке плагинов и расширений, которые обладают расширенными возможностями, но требуют специальных действий по установке от пользователя.

* JS не позволяет читать/записывать/изменять/копировать произвольные файлы в файловой структуре пользователя.
* Не имеет прямого доступа к операционной системе
* Java Script выполненный в одной вкладе не может обращаться к другим вкладкам и окнам.

**Тенденции развития**

С ростом популярности языка развиваютсья стандарты, что даем нам новые возможности. Много open-source проектов и бесплатных библиотек дают нам широкие возможности для создания как домашних проектов так и мощных приложений для крупных компаний. На данном этапе язык дает нам возможности построения full-stack проектов, которые полностью созданы на Java Script – от backend(серверная часть в которой сконцентрирована бизнес-логика приложения) части и заканчивая frontend( пользовательский интерфейс).

Фреймворки для JS – программные платформы которые определяют структуру программной системы и облегчающие разработку и объединение отдельных компонентов в большой полноценный проект. Простыми словами этот каркас на котором мы в дальнейшем выстраиваем проект.

Такие фреймворки могут стать «волшебной палочкой выручалочкой» для быстрой разработки веб-проектов. Это дает разработчику больше времени сконцентрироваться на создании сложных элементов и бизнес логике.

Преимущества которые мы получаем от использования фреймворков:

* Эффективность – хорошая структуризация готовых шаблонов и функция уменьшают количество времени, затрачиваемое на разработку.
* Безопасность – фреймворки изначально обеспечены встроенными системами безопасности и все готовые шаблоны прошли тестирование.
* Инкапсуляция – скрытие реализации многих функций от пользователя в целях дополнительной безопасности и удобства разработчика.
* Бесплатность – большинство фреймворков – это проекты с открытым исходным кодом. Это делает их бесплатными что снижает стоимость конечного продукта.

Лучшие Java Script фреймворки за 2017 год:

* Angular.js
* React.js
* Vue.js
* Ember.js
* Meteor.js

В заключении хотелось бы сказать, что несмотря на долгое развитие языка, сейчас с его помощью можно решать широкий круг задач и создавать любые приложения – от мобильных до высоконагруженных веб-приложений с возможностью работы и в офлайн и в онлайн режиме.

**4.Проектирование графического пользовательского интерфейса средствами COM-объектов**

Одним из главных преимуществ разработки с помощью объектно-ориентированных языков, таких как C++ и Java, Java Script является возможность эффективной инкапсуляции внутренних функций и данных. Это осуществимо именно благодаря объектной ориентированности этих языков. В объекте скрыты способы его реализации, а "наружу" предоставляется только хорошо определенный интерфейс, позволяющий внешним клиентам эффективно использовать функциональные возможности объекта. Технология СОМ обеспечивает эти возможности также с помощью определения стандартных способов реализации и предоставления интерфейсов СОМ-объекта.

***Component Object Model*** — объектная модель компонентов - это технологический стандарт от компании *Microsoft*, предназначенный для создания программного обеспечения на основе взаимодействующих компонентов, каждый из которых может использоваться во многих программах одновременно. Стандарт воплощает в себе идеи полиморфизма и инкапсуляции объектно-ориентированного программирования. Стандарт COM мог бы быть универсальным и платформо-независимым, но закрепился в основном на операционных системах семейства *Microsoft Windows.* В современных версиях *Windows COM* используется очень широко. На основе COM были реализованы технологии: *Microsoft OLE Automation, ActiveX, DCOM, COM+, DirectX*, а также *XPCOM*.

Использование виртуальных функций в базовом классе является центральным моментом в проектировании СОМ-компонентов. Определение абстрактного класса порождает таблицу, содержащую только открытые методы (т.е. интерфейс) класса. Класс IMath не содержит переменных-членов и функций реализации объекта. Его единственной задачей является порождение производного класса Math для виртуальной реализации методов интерфейса компонента.

В технологии СОМ доступ к компонентам обеспечивается только с помощью указателей на виртуальные таблицы. Таким образом, прямой доступ к конкретным данным компонента становится невозможным. Внимательно изучите наш пример. Он достаточно прост, но отражает ключевую концепцию СОМ — использование виртуальных таблиц для доступа к функциональным возможностям компонента. Наконец, СОМ-интерфейс представляет собой просто указатель на указатель виртуальной таблицы (virtual table, или Vtable) C++.

## Принципы работы COM

Основным понятием, которым оперирует стандарт COM, является *COM-компонент*. Программы, построенные на стандарте COM, фактически не являются автономными программами, а представляют собой набор взаимодействующих между собой COM-компонентов. Каждый компонент имеет уникальный идентификатор (*GUID*) и может одновременно использоваться многими программами. Компонент взаимодействует с другими программами через *COM-интерфейсы* — наборы абстрактных функций и свойств. Каждый COM-компонент должен, как минимум, поддерживать стандартный интерфейс «*IUnknown*», который предоставляет базовые средства для работы с компонентом. Интерфейс «*IUnknown*» включает в себя три метода: *QueryInterface,* *AddRef, Release.*

Windows API предоставляет базовые функции, позволяющие использовать COM-компоненты. Библиотеки MFC и, особенно, ATL/WTL предоставляют более гибкие и удобные средства для работы с COM. Библиотека ATL от Microsoft до сих пор остаётся самым популярным средством создания COM-компонентов. Но зачастую COM-разработка остаётся ещё довольно сложным делом, программистам приходится вручную выполнять многие рутинные задачи, связанные с COM (особенно это заметно в случае разработки на C++). Впоследствии (в технологиях COM+ и особенно .NET) Microsoft попыталась упростить задачу разработки COM-компонентов.

**5. Алгоритм функционирования программного обеспечения**

Что же такое программное обеспечение(ПО)? В общем и целом это набор команд и программ, которые выполняют следующие функции:

* управление работой компьютера
* дает возможность пользователю извлекать, изменять и передавать информацию
* служат для создания другого ПО

ПО это та вещь которая позволяет нам извлекать из вычислительных возможностей персональных компьютеров максимум пользы. От выполняемых функций программное обеспечение может делится на две группы: прикладное и системное ПО.

**Системное программное обеспечение**

Набор программ которые распределяют ресурсы компьютера и управляют его компонентами. Само оно не выполняет каких либо функция для пользователя и несет для практической ценности, но оно в свою очередь является базой для обеспечения работы прикладного ПО, абстрагируя детали аппаратной и микропрограммной реализации вычислительной системы от пользователя. Отнесение того или иного программного обеспечения к системному условно, и зависит от соглашений, используемых в конкретном контексте. Как правило, к системному программному обеспечению относятся операционные системы, утилиты, системы программирования, системы управления базами данных, широкий класс связующего программного обеспечения.

**Прикладное программное обеспечение**

Это ПО, которое было разработано пользователями или для пользователей с целью выполнения определённых задач пользователя. В зависимости от выполняемых задач они могут делиться по типам и сфере применения. По своему типу программное обеспечение можно разделить на:

* Общего назначения
* Развлекательного назначения
* Специального назначения
* Профессиональное ПО

Ввиду того что прикладное ПО позволяют комбинировать выполняемые задачи, сфера их применение довольна широка.

1. **Web Audi API для генерации звука**

Web Audio API – новинка, расширяющая возможности web приложений при работе со звуком. Это мощнейший инструмент, без которого Вам сложно будет обойтись в будущем при разработке современных игр и интерактивных веб приложений. API достаточно высокоуровневый, продуман до мелочей, самодостаточен, легок в освоении и особенно элегантно интегрируется в приложения, использующие WebGl и WebRTC.

Еще на ранних этапах развития web-а были предприняты попытки добавления способов по взаимодействию и манипуляции со звуком. Тогда не существовало строгих правил и стандартов и каждый производитель браузеров пытался создать свои способы – тег <bgsound> в Internet Explorer и <embed> для NetScape. По прошествии нескольких лет в браузеры была добавлена возможность установки сторонних плагинов и расширений. Работа с аудипотоками стала возможна благодаря таким плагинам как QuickTime, Flash и д.р.

Пионером в борьбе с тишиной без плагинов стал элемент <audio>, появившийся уже в первой спецификации html5. Он дает доступ к проигрыванию аудио файлов и стримов, контролировать воспроизведение, буферизацию и уровень звука. Кроме того, он прост в использовании и понимании. Сейчас он поддерживается всеми мобильными и десктопными браузерами (включая IE9), работает достаточно хорошо.

В начале важно понять, что элементы <audio> и web Audio API практически никак не связаны между собой. Это два независимых, самодостаточных API, предназначенных для решения разных задач. Единственная связь между ними состоит в том, что <audio> элемент может быть одним из источников звука для web Audio API.

Задачи, которые призван решать элемент <audio>:

* Простой аудио плеер
* Однопоточное фоновое аудио.
* Аудио подсказки, капчи и т.п.

Задачи, которые призван решать Web Audio API:

* Объемный звук для игр и интерактивных веб приложений
* Приложения для обработки звука
* Аудио синтез
* Визуализация аудио и многое, многое, многое другое…

Преимущества Web Audio API

* Абсолютно синхронное воспроизведение аудио (возможность проигрывать сотни семплов одновременно с разницей в миллисекунды, точно планируя начало и конец воспроизведения каждого из них)
* Возможность обработки звука с помощью десятков встроенных высокоуровневых блоков (фильтров, усилителей, линий задержки, модулей свертки, и т.д.)
* Богатые возможности для синтеза колебаний звуковой частоты с различной формой огибающей. (Можно написать простейший синтезатор за 10 мин)
* Работа с многоканальным аудио (Исходя из спецификации, API обязан поддерживать до 32 каналов аудио!!! Для справки: стерео – это 2 канала, Dolby Digital – это 5 каналов, самый навороченный Dolby TrueHD – 8 каналов, т.е на сегодняшний день у немногих пользователей дома есть звуковые карты с более чем 8-ю каналами :)
* Непосредственный доступ к временным и спектральным характеристикам сигнала (позволяет делать визуализации и анализ аудио потока)
* Высокоуровневое 3D распределение аудио по каналам, в зависимости от положения, направления и скорости источника звука и слушателя (что особенно круто при разработке объемных WebGL игр и приложений)
* Тесная интеграция с WebRTC (как источник звука можно использовать системный микрофон, подключить гитару или микшер. Вы также можете получить аудио из любого внешнего стрима, как, впрочем, и отправить его туда же)

1. **Анализ результатов решения поставленных задач**

В ходе выполнения курсовой работы были рассмотрены различные процессы и способы генерации цифрового звука, его обработке и преобразования из аналогового аудипотока в цифровой, что в современном мире очень актуально.

Был построен графический пользовательский интерфейс с выводом различных данных при генерации звука – частоты генерируемого цифрового звука и возможностью самостоятельно «рисовать» спектрограмму и на основе ее генерировать аудипоток.

Была написана программа по генерации звука используя Web Audio API и Java Script - встроенные возможности браузера и язык программирования, программы написанные на котором, возможно везде где для этого имееться интерпретатор.

Были получены навыки и открыты новые возможности современных стандартов HTML5 и JS, дающие нам эффективные способы по генерации звука и отрисовки элементво векторной графики прямо в web-браузере.

**Заключение**

С развитием современных технологий и стандартов языков программирования, возможности разработчиков существенно возросли. Сейчас они позволяют нам создавать невероятные вещи и решать широкий комплекс задач прикладного характера. Еще 15-20 лет мы такое не могли себе даже представить. И это рассматривая не только работу со звуком но и программирование в целом.

Кто в прошлом мог подумать что мы будем активно использовать нейронные сети, машинное обучение и компьютерное зрение(CV). Сейчас мы можем даже CV в браузере и распознавать объекты и лица попадающие в объектив веб-камер.

Фреймворки которые стали каркасами для будущих проектов позволили увеличить качество написанных программных продуктов и уменьшить скорость разработки , а так же итоговую стоимость продукта. Больше количество проектов с открытым исходным кодом дали нам бесплатные, но эффективные инструменты для решения многих задач прикладного характера.

Поэтому это накладывает на разработчика необходимость в изучении новых языков программирования, современных возможностей и следить за тенденциями развития новых технологий.

**Список использованных источников**

1. <https://learn.javascript.ru/?map>

2. Беллман Р. Динамическое программирование. — М.: Изд-во иностранной литературы, 1960.

3. <https://stackoverflow.com/questions>

4. https://developer.mozilla.org/en-US/docs/Web/JavaScript

5. https://habrahabr.ru/post/111308/

6. <http://html5.by/blog/audio/>

7. Дэвид Флэнаган. JavaScript. Подробное руководство. – O’Reilly, 2012.

**Приложение А**

**Листинг программного кода.**

(предварительно, будет еще дорабатываться немного)

let mimi=0,

inputText = document.querySelector('.text\_value'),

canvasElem = document.querySelector('.canvas'),

clearbtn = document.querySelector('.button\_clear'),

AudioContext = window.AudioContext || window.webkitAudioContext || window.mozAudioContext,

context = new AudioContext(),

ctx,

flag = false,

prevX = 0,

currX = 0,

prevY = 0,

currY = 0,

dot\_flag = false,

x = "black",

y = 2,

widthElem = window.innerWidth,

heightElem = window.innerHeight;

class Sound {

constructor(gainAmount,history,log,speed,fadeTime,elm,maxFreq){

this.gainAmount = gainAmount;

this.history = history;

this.log = log;

this.speed = speed;

this.fadeTime = fadeTime;

this.maxFreq = maxFreq;

this.elm = elm;

this.elm.addEventListener('mousedown', this.onMouseDown.bind(this));

this.elm.addEventListener('mouseup', this.onMouseUp.bind(this));

this.elm.addEventListener('mousemove',this.onMouseMove());

console.log("create a sound element");

};

onMouseDown(event){

console.log('md');

if (this.osc\_) {

return;

}

this.updatePointer(event);

this.osc\_ = this.createOscillator\_();

this.lastFreq = this.getLastFrequency();

this.osc\_.frequency.setValueAtTime(mimi, context.currentTime);

}

onMouseUp(){

console.log('mu');

this.deleteOscillator\_();

this.lastFreq = null;

}

onMouseMove(){

console.log('mm');

if (this.osc\_) {

return;

}

this.osc\_ = this.createOscillator\_();

this.lastFreq = this.getLastFrequency();

this.osc\_.frequency.setValueAtTime(mimi, context.currentTime);

// this.updatePage()

}

createOscillator\_(){

let gain = context.createGain();

gain.value = 0;

gain.connect(context.destination);

this.gain\_ = gain;

let osc = context.createOscillator();

osc.connect(gain);

gain.gain.linearRampToValueAtTime(this.gainAmount,

context.currentTime + this.fadeTime);

osc.start(0);

return osc;

};

deleteOscillator\_(){

if (this.osc\_) {

let endTime = context.currentTime + this.fadeTime;

this.gain\_.gain.linearRampToValueAtTime(0, endTime);

this.osc\_.stop(endTime);

this.osc\_ = null;

};

};

getLastFrequency(){

let x = this.lastX;

let y = this.lastY;

let percent = 1 - (y / this.height);

let nyquist = context.sampleRate/2;

if (this.log) {

percent = this.logScale\_(percent \* 1000, 1000) / 1000;

};

return percent \* nyquist;

};

updatePointer(event){

console.log("update\n", event.type)

event = event || {};

let type = event.type || '';

if (type.indexOf('mouse') == 0) {

this.lastX = event.pageX;

this.lastY = event.pageY;

} else if (type.indexOf('touch') == 0) {

this.lastX = event.touches[0].pageX;

this.lastY = event.touches[0].pageY;

} else {

this.lastX = null;

this.lastY = null;

}

};

// updatePage(evenet){

// this.getLastFrequency()

// event = event || {};

// this.lastX = event.pageX;

// this.lastY = event.pageY;

// currX = (window.Event) ? event.pageX : event.clientX + (document.documentElement.scrollLeft ? document.documentElement.scrollLeft : document.body.scrollLeft);

// currY = (window.Event) ? event.pageY : event.clientY + (document.documentElement.scrollTop ? document.documentElement.scrollTop : document.body.scrollTop);

// this.osc\_.frequency.value = (mimi/widthElem)

// console.log(this.osc\_.frequency.value,"\n",this.lastY,"\n",widthElem)

// // gainNode.gain.value = (currY/heightElem) \* maxVol;

// }

};

let initCanvas = ()=> {

ctx = canvasElem.getContext("2d");

canvasElem.width = widthElem;

canvasElem.height = heightElem;

console.log("init draw func")

canvasElem.addEventListener("mousemove", function (e) {

findxy('move', e)

}, false);

canvasElem.addEventListener("mousedown", function (e) {

findxy('down', e)

}, false);

canvasElem.addEventListener("mouseup", function (e) {

findxy('up', e)

}, false);

canvasElem.addEventListener("mouseout", function (e) {

findxy('out', e)

}, false);

}

let draw = () => {

ctx.beginPath();

ctx.moveTo(prevX, prevY);

ctx.lineTo(currX, currY);

ctx.strokeStyle = x;

ctx.lineWidth = y;

ctx.stroke();

ctx.closePath();

}

let clear = () => {

let m = confirm("Очистить холст?");

if (m) {

ctx.clearRect(0, 0, widthElem, heightElem);

}

}

let findxy = (res, e) => {

if (res == 'down') {

prevX = currX;

prevY = currY;

currX = e.clientX;

currY = e.clientY;

flag = true;

dot\_flag = true;

if (dot\_flag) {

ctx.beginPath();

ctx.fillStyle = x;

ctx.fillRect(currX, currY, 2, 2);

ctx.closePath();

dot\_flag = false;

}

}

if (res == 'up' || res == "out") {

flag = false;

}

if (res == 'move') {

if (flag) {

prevX = currX;

prevY = currY;

currX = e.clientX;

currY = e.clientY;

draw();

}

}

}

window.addEventListener('mousemove', ()=>{

mimi = event.pageY\*20;

inputText.value = mimi;

});

clearbtn.addEventListener('click', clear )

let some = new Sound(0.2,[],false,10,0.01,window,50000);

initCanvas();