- когда присутствуют отношения “один-ко-многим” (например, у пользователя есть возможность ввести несколько адресов, поэтому имеет смысл отделить адреса в отдельную таблицу);
- когда есть частая необходимость создать уникальный список из этих данных (например, для выпадающего меню, из которого будет выбирать пользователь);
- когда данные часто обновляются.

С целью оптимизации работы с данными принимать решение о денормализации рекомендуется лишь после проектирования нормализованной базы данных [2,3]. Необходимо проанализировать характер и частоту запросов и только после этого, при необходимости, аккуратно, с учётом сохранения целостности данных, произвести денормализацию [2, 3]. В таком случае её побочные эффекты снижаются до минимума, при этом обеспечивая увеличение быстродействия базы данных.

К недостаткам денормализации можно отнести:

1. большее потребление места на диске (из-за дублирования данных);
2. возможное замедление операций вставки, обновления и удаления данных;
3. аномалии данных (нарушение целостности данных): в какой-то момент данные могут измениться в нескольких местах, в таком случае нужно обеспечить корректное обновление их копий [3] (или в случае с расчётными данными – пересчитать расчётные значения, полученные из изменённых данных);
4. документация: денормализованную базу данных сложнее поддерживать, поэтому денормализацию базы данных следует подробно документировать (если возникнет необходимость поменять структуру базы данных, то нужно будет учитывать все предыдущие изменения) [3];
5. денормализация может потребовать написания большего количества кода [2], например, триггеров и процедур, обеспечивающих целостность данных, а если она проводится на существующей рабочей базе данных, то может понадобиться изменение существующих запросов и написание кода для обновления уже имеющихся записей [3].

Таким образом, залог быстрой и исправно работающей базы данных – это нахождение оптимального равновесия между нормализацией и денормализацией. Но началом и основой проектирования базы данных остаётся нормализация.

Список использованных источников:

1.Scott Selikoff , «Why too much Database Normalization can be a Bad Thing», November 19th, 2008.

2.Michelle A. Poolet, «Responsible Denormalization: How to break the rules and get away with it», 2002 Penton Media, Inc.

3.Блог компании Латера Софтвер, «Зачем нужна денормализация баз данных, и когда ее использовать», 9 апреля 2016

CHLORINE ENGINE И МИНИ-ИГРА НА НЕМ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Шпаков Н.И.

Жвакина А.В. – к.т.н., доцент

Игры все больше и больше заполоняют наш мир. Кто-то считает это лишь простой забавой, кто-то бизнесом, а для кого-то это жизнь или искусство. В любом случае, нельзя отрицать большого влияния этой отрасли на IT в целом. Развиваются технологии для обработки графики, звука, появляются новые технологии анимации и рендера, увеличивается мощность железа. Так или иначе, игры являются двигателем прогресса в мире компьютерных технологий.

В данной работе представлена игра, написанная на движке Clorineengine и сам движок, написанный с использованием библиотек SDL2 (окно и обработка событий), GLEW (для вытягивания функций OpenGL), OpenAL (библиотека для стереозвука), GLM (математическая библиотека), picoPNG (декодирование PNG файлов), freetype (преобразование текста в битмапы). В качестве языка программирования выбран C++ 11. Для сборки проекта используется технология CMake. Готовый проект с открытым кодом можно найти по ссылке на github: [https://github.com/ShpakovNikita/Chlorine-5].

Движок умеет обрабатывать пользовательский ввод, рисовать квадраты на поверхности (под углом также), проигрывать звуки с учетом модели дистанции, делать рэйкасты, проверять коллизии разными способами, отрисовывать простейший свет, рисовать GUI, динамический текст по шрифту, работать с анимациями спрайтов и многое другое. Отрисовка происходит посредством передачи данных о каждом полигоне в вершинный шейдер, а затем передачи его во фрагментный шейдер для дальнейшего вывода цвета каждого пикселя на экран. Звуки реализованы через модель источников и слушателя, где для каждого динамического объекта создается свой источник и для слушателя, «привязанного» к камере, идет просчет доходящего звука от источников. Игра же, по сути, в игровом цикле делает все эти действия последовательно, с учетом ООП модели.

Игра является относительно простым примером работы с движком, задействуя абсолютно все его возможности и реализуя множество алгоритмов совместно с функциями движка. Окружение в игре генерируется случайным образом с помощью алгоритма процедурной генерации подземелья, и динамически подбирает тайлы для создания изометрического эффекта. Для изометрического эффекта используется также zbuffer. Реализован также искусственный интеллект, который представлен системой конечных автоматов и

использует систему анимации и звука движка для реалистичного представления врагов.

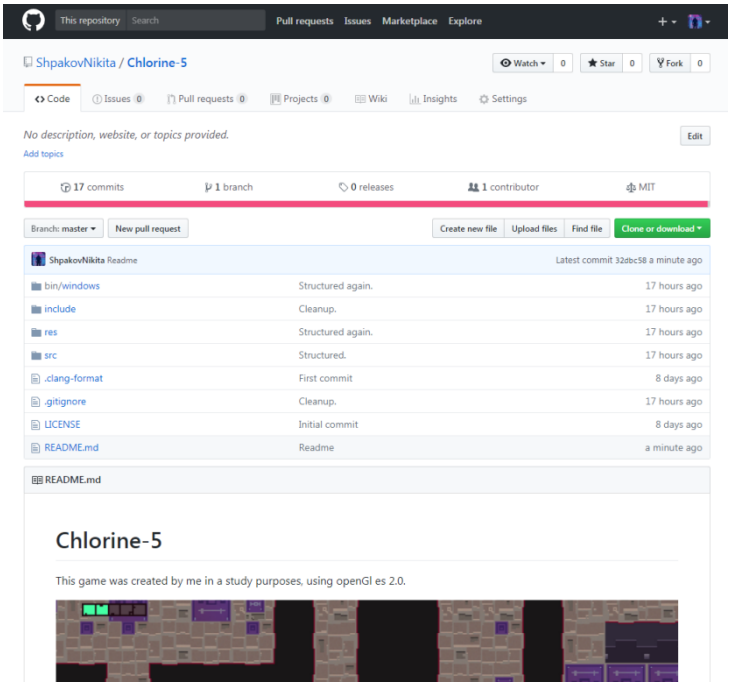

Рис. 1 – Проект на github

Основным преимуществом игрового движка является его легковесность. В нем присутствуют только самые основные функции, которых достаточно для создания полноценной 2д игры. Также движок может быть легко кастомизирован из-за подробного описания каждой функции и своего небольшого размера. Другим преимуществом является полная открытость исходного кода движка, который большинство других движков до сих пор скрывают. Однако у движка имеются недостатки, связанные с архитектурой, отсутствует возможность автоматического портирования проекта на другие платформы и пользователю придется делать это вручную. Устранение перечисленных проблем позволит игре стать более универсальной и конкурентоспособной.

Список использованных источников:

1. Материалы для обучения OpenGl. [Электронный ресурс] – Режим доступа: https://learnopengl.com. − Дата доступа: 13.03.2018.

2. Документация OpenGL. [Электронный ресурс] – Режим доступа: http://www.khronos.org. − Дата доступа: 13.03.2018.

3. Официальный сайт OpenAL с доступной pdf документацией. [Электронный ресурс] – Режим доступа: http://www.openal.org. − Дата доступа: 13.03.2018.

4. Документация SDL1, SDL2. [Электронный ресурс] – Режим доступа: http://wiki.libsdl.org/FrontPage. − Дата доступа: 13.03.2018.