

Сборка

Сборка производилась на Win10 x64 с использованием C++17 & CMake & Ninja & MinGW 8.1 x64 в среде QtCreator.

Также проверялась сборка на VS 2022 (сгенерированный из CMakeLists.txt) & C++20 && MSVC 143.

CMakeLists.txt ориентирован на Windows, но под Linux, в теории, тоже должно работать, если все библиотеки SDL установлены, проверить возможности нет.

Исходный код лежит в директории **/src**.

В директориях **/SDL2** и **/SDL2_ttf** лежат библиотеки SDL.

В директории **/images** находятся текстуры для ячеек барабанов.

Запуск

В директории **/build** находится исполняемый файл **/SDL2Test.exe** с библиотеками **SDL2.dll** & **SDL2_ttf.dll**.

Также необходимо убедиться в наличии директории **/images** с файлами текстур и файла шрифта **JetBrainsMono-Regular.ttf** в директории с исполняемым файлом.

Запуск проверялся также на Win10 x64.

Описание исходного кода

Далее будет приведено краткое описание основных компонентов программы.

MainWindow	
bool init ();	Инициализирует все компоненты SDL и игровой логики
bool loop ();	Запускает бесконечный цикл рендера и обработки событий
void startDrums ();	Запуск прокрутки барабанов
void render (Stuff::FrameInfo frameInfo);	Рендер всех игровых компонентов

Button	
Button (SDL_Renderer* renderer , const std::string & file , const SDL_Rect & pos , std::function< void(void) > & callback);	Конструктор, принимающий на вход <ul style="list-style-type: none">- рендер- файл для текстуры- положение кнопки- функцию обратного вызова

void render(float timeStep);	Отрисовка кнопки на экране. При отрисовке меняется прозрачность текстуры от minTransparency до maxTransparency и в обратную сторону.
void handleEvent(const SDL_Event &event);	Метод для определения нажатия кнопки.
void Button::setEnabled(bool enable)	При enable == true устанавливается максимальная прозрачность, при enable == false устанавливается минимальная прозрачность и отключается анимация мигания

Cell	
Cell(SDL_Renderer *renderer, const std::string &file, const SDL_Rect &currentRect, const SDL_Rect &borders);	Конструктор для создания ячейки в барабане, принимающий: <ul style="list-style-type: none"> - рендер - файл текстуры - изначальное положение ячейки - границы области, внутри которой она может перемещаться
void render();	Метод отрисовки ячейки барабана. Если установлен флаг isSplitted, то отрисовывается также "дополнительная" текстура fadeTexture, представляющая собой появляющуюся часть ячейки сверху.
void move(float step);	Метод для перемещения ячейки по вертикали. Проверяет выход за границы ячейки и устанавливает соответствующие размеры ячейки при их пересечении. Перемещение вверх не поддерживается.
void reset(int y);	Сбрасывает размеры основной текстуры до изначальных и помещает ячейку по вертикальной оси в положение 'y'

Drum	
Drum(SDL_Renderer *renderer, const std::vector<std::string> &files, const SDL_Point &pos, const SDL_Point &cellSize);	Конструктор для создания одного барабана, принимающий: <ul style="list-style-type: none"> - рендер - вектора файлов текстур для

	<p>каждой ячейки</p> <ul style="list-style-type: none"> - положение барабана - размер ячеек
<code>void render(float timeStep);</code>	<p>Метод отрисовки ячеек барабана. При “вращении” перемещает все ячейки на шаг moveStep. При окончании перемещения (когда заканчивается “дистанция перемещения” targetShufflingDistance, останавливает вращение и фиксирует положение ячеек.</p>
<code>void startShuffle(float seconds);</code>	<p>Запускает вращение барабана. Генерирует случайные результирующие индексы положения ячеек и скорость вращения, на вход принимает количество секунд, которое необходимо вращать барабан.</p>

FPSCounter	
FPSCounter (SDL_Renderer *renderer, const std::string &fontFile, SDL_Rect pos, SDL_Color color);	<p>Конструктор для объекта, отображающего количество кадров в секунду и принимающий:</p> <ul style="list-style-type: none"> - рендер - файл шрифта - положение - цвет текста
<code>void render(float fps);</code>	Отображает значение fps

Texture	
static std::unique_ptr<Texture> loadTexture (SDL_Renderer *renderer, const std::string &file, SDL_BlendMode blendMode = SDL_BLENDMODE_BLEND);	<p>Фабричный метод для загрузки текстуры, принимающий:</p> <ul style="list-style-type: none"> - рендер - файл текстуры - режим текстуры
static std::unique_ptr<Texture> loadTextTexture (SDL_Renderer *renderer, const std::string &text, TTF_Font *font, SDL_Color color);	<p>Фабричный метод для загрузки текстовой текстуры, принимающий:</p> <ul style="list-style-type: none"> - рендер - текст для отрисовки - шрифт - цвет

<code>void renderTexture(const SDL_Rect &dst);</code>	Метод для отрисовки текстуры в положение dst
<code>void renderTexture(const SDL_Rect &src, const SDL_Rect &dst);</code>	Метод для отрисовки части текстуры src в положение dst
<code>void updateTransparency(Uint8 a);</code>	Метод для изменения степени прозрачности текстуры

TextureLoader	
<code>static SDL_Texture* loadBMP(SDL_Renderer *renderer, const std::string &file)</code>	Фабричный метод для загрузки текстуры из .bmp файла, принимающий: <ul style="list-style-type: none"> - рендер - файл текстуры
<code>static SDL_Texture* loadTextTexture(SDL_Renderer *renderer, const std::string &text, TTF_Font *font, SDL_Color color)</code>	Фабричный метод для создания текстовой структуры, принимающий: <ul style="list-style-type: none"> - рендер - текст для отображения - шрифт - цвет текста

FPSTimer	
<code>FrameInfo getFrameInfo();</code>	Метод для подсчета количества кадров в секунду и времени кадра. Необходимо вызывать при каждой итерации цикла отрисовки.

UniformRandom	
<code>static int uniformInt(int min, int max);</code>	Метод для генерации числа от min до max на основе равномерного распределения

Улучшения & недостатки

- Не учитываются размеры текстуры и требуется размер 100x100 или менять в исходном коде, иначе ячейки “плывут”
- FPSTimer выделить в класс-одиночку для использования единого инстанса во всех участках программы
- Отсутствие проверок на неадекватные значения количества ячеек и т. д.

- Вместо исключения при каких-либо ошибках не будут отрисовываться объекты, которые не смогли загрузиться без информации об этом, кроме логов -> MessageBox
- Классы для работы с текстурами представляют только минимум функций, который нужен по заданию
- Поддерживается только .bmp из коробки SDL2, SDL2_img не подключался из-за отсутствия необходимости
- Отсутствует ручное ограничение частоты кадров - для этого установлен флаг `SDL_RENDERER_PRESENTVSYNC`, синхронизирующий принудительно частоту с монитором
- Не корректируется размер области с текстом о количестве кадров в секунду в зависимости от количества цифр
- Местами может хромать форматирование кода из-за не до конца настроенного автоформатирования
- Сборка на Linux может потребовать корректировки CMakeLists.txt