Документація до семестрового проекту  
з дисципліни Основи Об’єктна орієнтоване програмування

Студента факультету   
комп'ютерних наук та кібернетики  
Групи ІПС-22  
Дехтяренка Михайла Віталійовича

Минулого семестру я розробляв гру жанру tower defense. Тоді у звіті писав, що більше не планую займатись нізькорівневою розробкою ігор, використовуючи графічні бібліотеки та краще надам перевагу ігровим рушіям як Unity або Unreal Engine. Але я отримав пораду спробувати бібліотеку SDL2.

Тому було вирішено написати проект на SDL2. Вибір пав на жанр шутер, а саме як старі Doom, Wolfenstein та інші. Щоб не ускладнювати процес розробки більш комплексною системою 2.5 вимірного простору, де гравець міг би змінювати свою позицію відносно осі Z, було вирішено реалізувати простий і приближений до оригінального Wolfenstein алгоритм Raycaster.

Raycaster - є алгоритмом, який широко використовується в розробці ігор для рендерингу тривимірних сцен з використанням двовимірних карт. Основна ідея полягає у випромінюванні "променів" з точки зору гравця і визначенні, де ці промені перетинаються з об'єктами сцени, враховуючи кут та відстань.

Початковою ідеєю гри була реалізація швидкого проходження, щоб потім результати вносити в базу даних в облаці (розглядалась Mangodb), а потім виводилась на захощеному сайті, але від цієї ідеї прийшлось відмовитись через нехватку досвіду та часу на реалізацію.

Загалом в проекті реалізовано

- Рух гравця

- Можливість стріляти

- Звук під час пострілу

- Рух ворогів за гравцем

- Броунівський рух (через мультипоточність, щоб кожен ворог рухався незалежно один від одного)

- Зміна рівнів, коли всі вороги помирают (поки реалізовано 2 рівня)

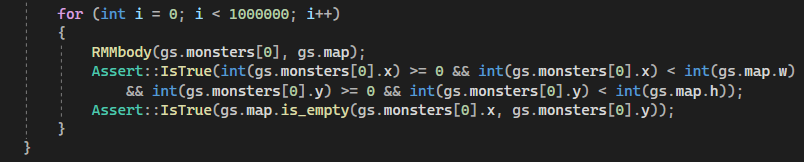
- Смерть гравця

- Смерть ворогів

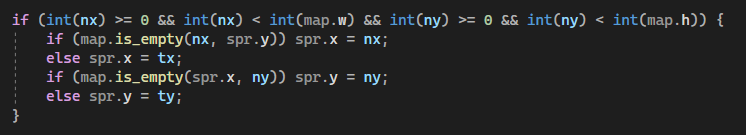
Під час розробки проекту була опанована робота з графічною бібліотекою SDL2 та її похідними: SDL\_MIXER – для роботи зі звуком та SDL\_TTF - для виведення тексту (в контексті грі – реалізації таймеру)

Також в проекті було реалізовані 3 юніт тести через naive unit tests в середовищі Visual Studio.

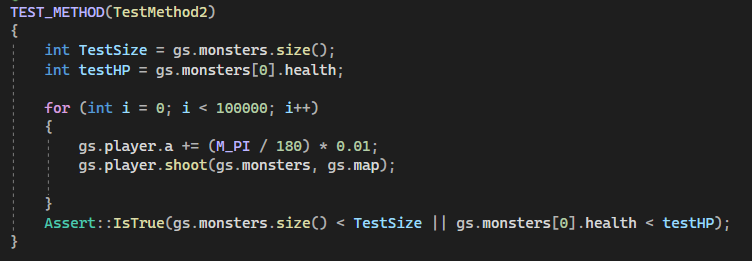
Перший перевіряв чи може виходити ворог за стінки або межі карти. Завдяки цьому тесту було помічено, що від час реалізації броунівського руху не була передбачена можливість ворога вихода за рівень.

Сам юніт тест:  


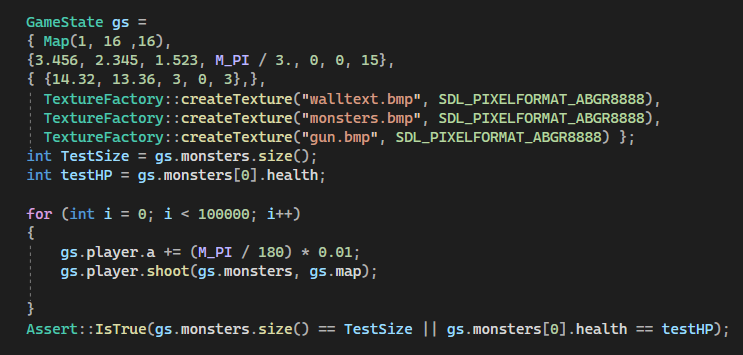
Код, який з’явився для того, щоб юніт тест завжди був позитивний



Другий юніт тест перевіряє коректність роботи системи пострілу, нанесення урону ворогу та його вбивства. «Гравець» робить сто тисяч обертів, і, якщо ворог помер(кінцевий масив менше за початковий) або здоров’я ворога менше за початкове, то тест вважається пройденим



Третій тест має схожий алгоритм роботи, але перевіряє чи не проходять кулі скрізь стінки.

 Для цього ми розташували ворога на іншому кінці мапи та знову обертаємо «Гравця» навколо своєї осі. Якщо кількість ворогів або здоров’я ворога не зміниться, значить тест пройде

Також під час написання юніт тестів прийшлось шукати рішення проблем в лінкером, що потім дозволило оптимізувати файли проекту

Як висновок можу сказати, що працювати з бібліотекою SDL2 меня сподобалось на багато більше, ніж з OpenGL. Навчився працювати з лінкером і покращив свої навички роботи з сторонніми бібліотеками. Також сподобалось використання патернів, оскільки вони дозволяють систематизувати код та зробити його більш гнучким ( в контексті більших проектів).