Indywidualny projekt programistyczny

Mateusz Połeć

Cele projektu

Projekt miał na celu stworzenie prototypu funkcjonalnego silnika do tworzenia gier 2D, opartego na bibliotece SFML i OpenGL, dodatkowo uwzględniając dodatkowe biblioteki, pomagające osiągnąć pokrewne cele. (Wczytywanie mapy, działania na plikach o różnych formatach np. **JSON**)

Co udało się osiągnąć?

- Obsługa wielu klawiszy wciśniętych na raz,
- Stworzenie maszyny stanów do obsługi scen,
- Stworzenie pełnego systemu obiektów i komponentów (Podobnie, jak w Unity),
- Stworzenie systemu zasobów, (Tekstury wczytane raz zostają dodane do klasy, która je obsługuje i przypisuje do odpowiednich obiektów),
- Stworzenie silnika animacji, (Animacja wycina odpowiednie kwadraty/prostokąty z plików o formacie .png),
- Stworzenie silnika wczytywania mapy, (Sam szkielet - celem było zrealizowanie tego zadania samodzielnie, bez wspierania się bibliotekami i dodanie możliwości wczytywania warstw wielowątkowo, co znacznie wpłynęłoby na wydajność),
- Stworzenie systemu kolizji, (Kolizje zapisywane są na mapie za pomocą odpowiedniej warstwy, a silnik odpowiednio adaptuje obiekty kolizji),
- Stworzenie systemu kamer, (Kamera nigdy nie wychodzi poza obszar mapy i jest dynamiczna - przesuwa się wraz z ruchem obiektu),
- Stworzenie menu wejściowego,
- Stworzenie opcji, (Możliwość dobierania rozdzielczości i ustawienia VSync),
- Stworzenie systemu logów do wykrywania błędów i debugowania kodu.

Czego nie udało się osiągnąć?

- Debugowanie systemu kolizji (Czasami, kiedy kolizji wokół gracza jest za dużo, silnik nie potrafi odpowiednio utworzyć priorytetu, która kolizja jest ważniejsza i gracz może wpaść w obiekt i mieć kłopoty, aby się z niego wydostać),
- Stworzenie systemu projectiles (System miał zakładać tworzenie różnych, dostosowanych do mapy obiektów i animacji (jak np. **krople deszczu**),
- Stworzenie systemu zadań (**questów)**,
- Stworzenie systemu dialogów,
- Stworzenie silnika **scen przerywnikowych** (czasami, kiedy gracz w grze skończy dane zadanie wyświetla się **cut-scene**, który ma chociażby pokazać efekty zrealizowanego zadania),
- Stworzenie systemu shaderów (Zdecydowanie pomogłyby w tworzeniu nowych efektów graficznych, bez ingerencji programisty w tekstury, mapę),
- Częściowo stworzenie grywalnego demo gry.

Zastosowane technologie

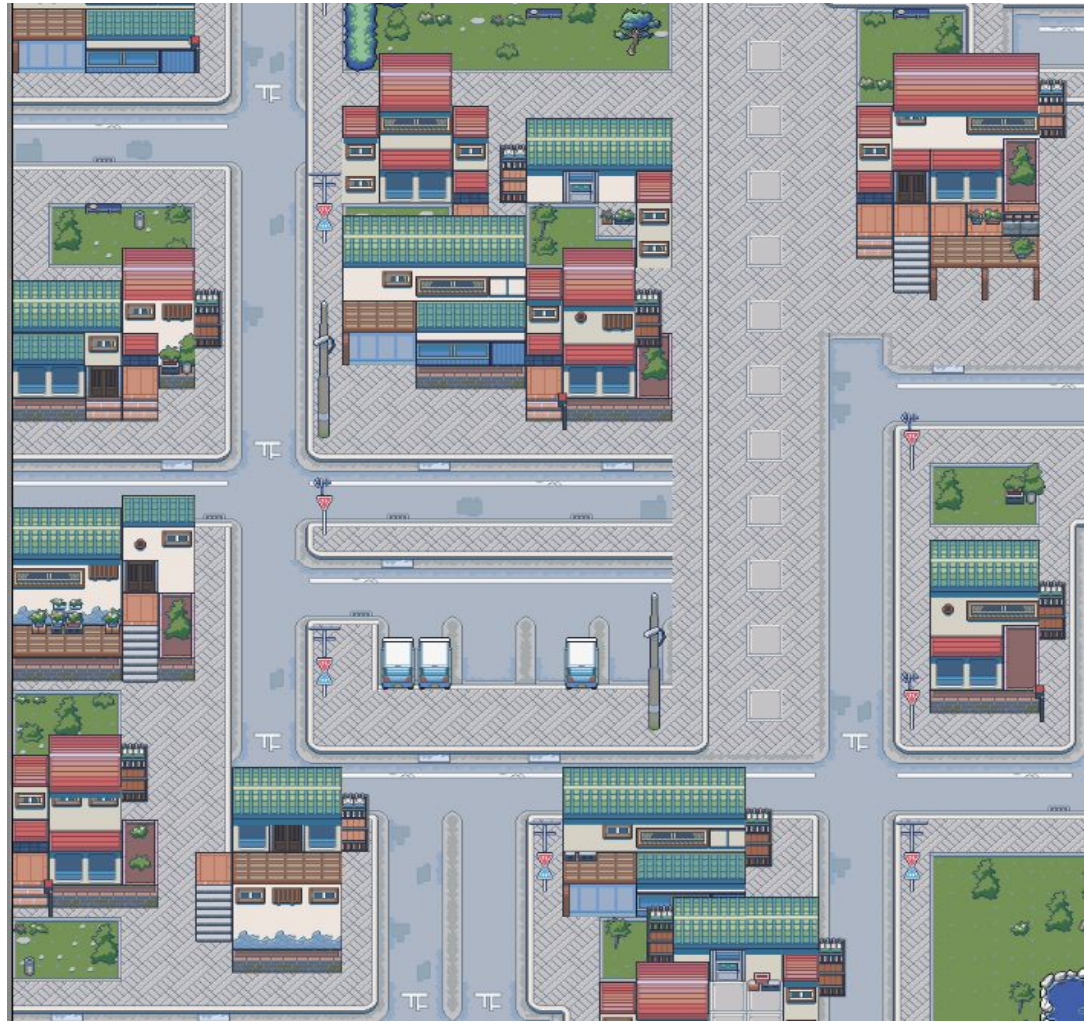
Cały projekt realizowany był na języku C++ w standardzie 17. W skład silnika wchodzą biblioteki takie jak:

- SFML (Wersja 2.5.1),
- STL (Użycie kontenerów i algorytmów),
- tmxlite (Biblioteka znacznie ułatwiająca wczytywanie mapy do gry)

Przykładowe lokacje w demo gry

Ogród Zen:

Miasto 1



Przykładowe lokacje demo gry:

Miasto 2:

Miasto 3:

Przykładowe zdjęcie z rozgrywki na demo gry:

Czego nie udało się zrealizować w przykładowej grze?

Podpunkt ten jest ściśle powiązany z samym silnikiem - w grze zdecydowanie nie udało się zrealizować podstawowego jej celu, czyli uczynić, aby gra była funkcjonalna i grywalna.

Żeby dokończyć fabularny aspekt gry, należałoby dokończyć systemy dialogów, zadań i przerywników scen.

Napotkane problemy

1. Zdecydowanym problemem było odpowiednie połączenie wszystkich klas i funkcjonalności tak, aby kod był czytelny i umożliwiał elastyczność dla programisty, który chciałby tworzyć grę - **Rozwiązany**, za pomocą zastosowania różnych sztuczek (programowania generycznego, dokumentacji kodu i stosowania wzorców projektowych),

2. Drugim napotkanym problemem był system kolizji, który do teraz wymaga debugowania - **Częściowo rozwiązany**, kolizje udało się zrealizować za pomocą matematycznych działań - obliczania wektora od środka obiektu do gracza, który uniemożliwia wejście graczowi na obiekt kolizji,

3. Trzecim napotkanym problemem było wczytywanie mapy - **Częściowo rozwiązany,** zastosowanie odpowiedniej biblioteki i połączenie jej z silnikiem rozwiązało różne błędy związane z mapą, jednak wciąż wydajność pozostawia wiele do życzenia - mapy, które mają więcej niż ~500 000 pikseli potrafią zabrać sporo czasu w trakcie kompilacji,

4. Ostatnim dość poważnym problemem było stworzenie menu z opcjami - **Rozwiązany,** głównym problemem była tutaj responsywność eventów, na które gracz może kliknąć - zmiana rozdzielczości powodowała, że przyciski się przemieszczały, jednak ich **ClickEvent** pozostawał w miejscu sprzed zmiany rozdzielczości. Udało się to rozwiązać za pomocą odpowiedniego skalowania całości przy drobnych zmianach rozdzielczości.

Ciekawostki powiązane z projektem

1. Projekt na ten moment ma **3470** linii kodu, oraz **1273** komentarze, pomagające w dokumentacji dla programisty, w tym:
   - 42 pliki nagłówkowe (.hpp),
   - 34 pliki źródłowe (.cpp)

2. Przedstawiona mapa demo, ma wielkość **100**x**100** przy przyjętych 32 pikselach na jeden kafelek, daje to mapę o wielkości 320 000 pikseli.

3. Pierwsze uruchomienie IDE i pierwsza kompilacja zajmuje około **16 sekund**, każda kolejna około **6 sekund**.

DZIĘKUJE ZA UWAGĘ :)