Projekt z C - gra o budowaniu miasta

Artur Jankowski

1 Instalacja

Będąc w folderze gry(zawierającym plik Makefile) należy użyć polecenia make all. Pliki wykonywalne znajdują się w build/
make clean - czyści pliki stworzone w trakcie kompilacji

2 Zastosowane narzędzia

Biblioteka SFML, C++11, cbp2make

3 Obsługa

- Wyjście z gry - Escape
- Zapisanie gry - F5 | T
- Wczytanie gry - Space
- Pełny ekran - F11
- Przybliżanie/Oddalanie - Scroll myszy | (+/-)
- Poruszanie kamerą - strzałki | klawisze WSAD | zbliżanie myszy do krawędzi ekranu (im bliżej, tym szybciej porusza się kamera)
- Zaznaczanie - kliknięcie lub kliknięcie i przeciągnięcie myszy
- Zmiana tekstury - klawisze 0 - łąka | 1 - las | 2 - sklepy | 3 - osiedla | 4 - fabryki | 5 - drogi | 6 - woda

4 Klasy

4.1 StateMachine

Stos obsługujący aktualny stan gry.
Funkcje:

- void pushState(State* stateRef, bool isChanging = true)
  Przyjmuje stan, który chcemy dodać
  isChanging (true - zamieniamy stan na górze stosu, false - dodajemy na górę stosu)
- void popState()
- void applyPendingChanges() - przetwarza zmiany na stosie

4.2 MediaHandler

Wczytuje tekstury i czcionki i mapuje je do tekstu-klucza, by pozwolić na wygodne używanie nazw tekstur zamiast długich ścieżek do pliku. Funkcje:

- void loadTexture(const std::string texName,const std::string filepath) - nazwa i ścieżka np. m_data->graphics.loadTexture(”commercial”, ”assets/Tile/commercial.png”);
- sf::Texture &getTexture(const std::string texName)
- void loadFont(const std::string fontName,const std::string filepath)
- sf::Font &getFont(const std::string fontName)

4.3 Animation

Obsługuje animacje. Konstruktor:
AnimationComponent(int maxID, float animSpeed, int width, int height)

- maxId - ile jest klatek animacji (numerowane od zera)
- animSpeed - domyślnie 6.0f, im większa wartość, tym szybsza animacja
- width, height - szerokość i wysokość pojedynczej kraty w pixelach

public:
sf::IntRect m_currentBound - obecna pozycja animacji

4.4 Button

Przycisk z tłem, wykrywający najechanie myszką i kliknięcie. Funkcje:

- void update(const sf::Vector2f mousePos) - sprawdza warunki na podstawie pozycji myszy
- void draw(sf::RenderWindow& window)
- void setTexture(sf::Texture& texture)
- void updatePosition(float x, float y, float width, float height) - pozycja i rozmiar przycisku
- bool isHovering() - sprawdza, czy kursor jest nad przyciskiem
- bool isPressed() - sprawdza, czy wciśnięto przycisk

Klasa zawiera dwie zmienne publiczne:

- MouseState m_mouseState - VOID | HOVERING | PRESSED
- sf::Color m_backgroundRectColor - kolor tła przycisku;

4.5 World

Klasa obsługująca mapę gry. Mapa zbudowana jest z wektora krat (instancje klasy Tile). Funkcje:

- void load(const std::string& filepath,std::map<std::string, Tile> &m_tileRefResources)
- void save(const std::string& filepath)
- void draw(sf::RenderWindow& window, float dt)
- void replaceTiles(Tile tile) - zamienia wszystkie obecnie zaznaczone Tile na podaną w argumencie
- void clearSelected() - usuwa zaznaczenie
- void selectArea(sf::Vector2f mousePosBeg, sf::Vector2f mousePosEnd) - zaznacza Tile na podstawie współrzednych (po wykonaniu funkcji MapPixelToCoords) kursora w dwóch różnych momentach
- void rotateRoads() - aktualizuje teksturę drogi by dopasowywała się ona do do otoczenia

4.5.1 Tile

Klasa definiująca pojedynczą kratę
Tile(sf::Texture& texture, TileType type,const unsigned int tileSize, int maxId, float animationSpeed, int cost)

- tileSize - rozmiar w pixelach
- maxId - ile jest klatek animacji (numerowane od zera)
- animationSpeed - domyślnie 6.0f, im większa wartość, tym szybsza animacja
- cost - koszt budowy kraty

Enum class TileType przyjmuje wartości:
| EMPTY | FOREST | COMMERCIAL | RESIDENTIAL | INDUSTRIAL | ROAD | WATER | MINE

4.6 GameLoop

Główna pętla programu. Przechowuje ważną strukturę GameData jako shared_ptr
. GameLoop(int width, int height, std::string title) - wymiary i tytuł okna gry
GameData:

- MediaHandler graphics - menadżer zasobów
- StateMachine machine - stos stanu gry
- sf::RenderWindow window - aktualne okno gry
  bool isFullScreen = false; - czy jest pełny ekran
  std::string title - tytuł paska gry
- std::map<std::string, Tile> m_TileMap - mapuje nazwę kraty (TileType, z małych liter) do jej konstruktora

4.7 State

Klasa, z której dziedziczą inne stany. Posiada funkcje wirtualne:
init(), handleInput(), update(float dt), draw(float dt), stop(), resume()

4.7.1 MainMenuState

Klasa obsługująca główne menu

4.7.2 MainGameState

Klasa obsługująca stan gry, w którym widzimy mapę

Spis treści