OBRAZKOWE UKŁADANKI

Jakub Kaleciński, Łukasz Turyński, Jakub Koch

1. Opis projektu

Należało napisać grę, w której grający ma ułożyć kompletny obrazek z wcześniej pomieszanych części. Rozgrywka trwa do poprawnego ułożenia obrazka lub gdy gracz przerywa grę. W pierwszej wersji podzielono obrazek na kwadraty. Na początku gry kwadraty przemieszano, niektóre obrazki mogły dodatkowo być odbite symetrycznie zarówno w pionie, jak i w poziomie. Gdy gracz klika pomiędzy sąsiednie kwadraty, to zamieniają się miejscami. Gdy kwadraty sąsiadują ze sobą poziomymi krawędziami, to przy zamianie następuje odbicie wzdłuż osi poziomej, jeżeli zaś kwadraty sąsiadują pionowymi krawędziami, to każdy z kwadratów jest odbity wzdłuż osi pionowej. W drugiej wersji brakuje jednego klocka, dzięki czemu składowe klocki mogą przemieszczać się względem siebie.

2. Założenia wstępne przyjęte w realizacji projektu

Aby spełnić wymagania podstawowe, program ma wczytać jeden z losowo jeden z kilku przygotowanych wcześniej obrazów, dzielić go na kwadraty a następnie w taki sposób je pomieszać, aby możliwe było ułożenie obrazka. Pod klawiszem tabulacji powinna znaleźć się opcja przywoływania prawidłowo ułożonego obrazka, tzn. wciśnięcia klawisza [Tab] powinno spowodować wyświetlenie na planszy gry poprawnie ułożonego obrazka. Ponowne wciśnięcie klawisza [Tab] powinno przywracać poprzedni układ planszy gry. Program sprawdza również po każdym ruchu czy użytkownik ułożył puzzle prawidłowo. Podczas zamiany miejsc kwadraty mogą przemieszczać się skokowo.

W wersji rozszerzonej program powinien pozwalać na wybór liczby kwadratów na jakie obrazek ma być podzielony. Program powinien również zliczać liczbę operacji wykonanych przez gracza oraz czas od rozpoczęcia rozgrywki. Dla każdego obrazka program ma zapamiętywać nicka grającego oraz jego czas i liczbę operacji. Dodatkowo w pierwszej wersji gry można było wprowadzić obracanie kwadratów. Wówczas kliknięcie na skrzyżowaniu krawędzi spowodowałoby przesunięcie czterech sąsiadujących kwadratów zgodnie z ruchami wskazówek zegara, a każdy z kwadratów obróciłby się wokół własnego środka symetrii w tym samym kierunku. W drugiej wersji gry należało wprowadzić jej dodatkową odmianę. Z układanki nie był wyrzucany żaden klocek. Gra pozwalała wtedy na przesuwanie całych wierszy w prawo lub w lewo oraz całych kolumn w górę lub w dół. Podczas zamiany miejsc kwadraty miały przemieszczać się w sposób płynny.

3. Analiza projektu

Specyfikacja danych wejściowych:
Dane wejściowe to jedno z losowo wczytanych zdjęć w formacie png.

Opis oczekiwanych danych wejściowych:
Program niczego nie zwraca, jedynie po wyborze wersji gry na ekranie jest wyświetlany wymieszany obrazek podzielony na mniejsze kwadraty.

Zdefiniowanie struktur danych:
Za przechowywanie pojedynczych kwadratów odpowiadają zmienne dostępne w bibliotece SFML, wszystkie podzielone kwadraty zaś są przechowywane w macierzy _board w stworzonej przez nas klasie Board.

Specyfikacja interfejsu użytkownika:
Po naciśnięciu klawisza [1] podzielone kwadraty są przemieszane i gracz rozpoczyna rozgrywkę na zasadach przedstawionych w 1. wersji, po kliknięciu [2] kwadraty są odpowiednio pomieszane oraz gracz zaczyna grę w 2. wersji. Po naciśnięciu [Tab] użytkownik może wyświetlić poprawnie ułożony obrazek, a ponowne kliknięcie [Tab] przywraca do gry.

Decyzja o wyborze narzędzi programistycznych:
Skorzystano z biblioteki SFML, a sam projekt napisano za pomocą Visual Studio Code.

4. Podział pracy, analiza czasowa , wyodrębnienie i zdefiniowanie zadań:

Jakub Kaleciński stworzył klasy obsługujące podział obrazka na mniejsze kwadraty oraz ich poprawnym wyświetlaniem na ekran, Łukasz Turyński odpowiadał za obsługę 1. wersji gry oraz napisanie dokumentacji, a Jakub Koch zajął się prawidłowym działaniem 2. wersji. Praca nad projektem każdemu z nas zajęła kolejno: 4 godziny, 6 godzin oraz 7 godzin.

5. Opracowanie i opis niezbędnych algorytmów

W programie nie zastosowano skomplikowanych algorytmów. Dzięki sprawdzaniu, który klawisz został wciśnięty lub w której wersji gry obecnie znajduje się użytkownik gra działa prawidłowo.

6. Kodowanie

Stworzono cztery klasy:

- Square, która reprezentuje pojedynczy kwadrat z podzielonego obrazka,

- Board, która przechowuje wszystkie podzielone kwadraty i odpowiada za ich prawidłowe wyświetlanie,
- Version1, która obsługuje zdarzenia związane z pierwszą wersją gry,
- Version2, która obsługuje zdarzenia związane z drugą wersją gry.

Klasa Square reprezentuje pojedynczy podzielony kwadrat. Klasa przechowuje następujące zmienne:

- int id,
- bool _flip_hor, wartość true oznacza, że kwadrat jest obrócony względem poziomej osi,
- bool _flip_ver, wartość true oznacza, że kwadrat jest obrócony względem pionowej osi,
- sf::Sprite _sprite, przechowuje podzielony obrazek.

Metody klasy Square:

- void draw(sf::RenderTarget& target, sf::RenderStates states) const override, przeładowuje metodę z podstawowej klasy abstrakcyjnej sf::Drawable,
- void change_position(const unsigned i, const unsigned j), zmienia pozycję danego kwadratu; po wywołaniu metody kwadrat znajduje się na pozycji [i,j] w macierzy _board,
- void flip_vertical(), odbija kwadrat wzdłuż pionowej osi,
- void flip_horizontal(), odbija kwadrat wzdłuż poziomej osi,
- int check() const, zwraca id kwadratu gdy kwadrat nie jest obrócony, w przeciwnym wypadku zwraca -1,
- static sf::Vector2i get_ground_zero(const unsigned i, const unsigned j), zwraca pozycję lewego górnego rogu danego fragmentu tekstury na podstawie przekazanych indeksów i,j,
- static sf::Vector2f get_position(const unsigned i, const unsigned j), zwraca położenie kwadratu na ekranie na podstawie indeksów i,j

Klasa Board, która przechowuje wszystkie podzielone kwadraty i odpowiada za ich prawidłowe wyświetlanie. Przechowuje następujące zmienne:

- const unsigned _height, _width, ilość kwadratów odpowiednio w jednej kolumnie i wierszu,
- std::vector<std::vector<Square>> _board, macierz przechowująca wszystkie kwadraty,
- enum class Mode {normal, preview}, normal oznacza, że gra jest w trybie, w którym gracz może zamieniać kwadraty, preview zaś pozwala przywrócić początkowy układ obrazka.

Dodatkowo przechowywane są zmienne konieczne do prawidłowego wyświetlenia kwadratów:

- sf::Texture _texture,

- sf::RenderTexture _render_texture
- sf::Sprite _sprite.

Metody klasy Board:

- void draw(sf::RenderTarget& target, sf::RenderStates states) const override, przeładowuje metodę z podstawowej klasy abstrakcyjnej sf::Drawable,
- void switch_mode(), przełącza pomiędzy trybem normal, a preview,
- virtual bool solved() const, zwraca true gdy obrazek jest prawidłowo ułożony, w przeciwnym razie false,
- void move(sf::Vector2i mouse_position), jeśli gra znajduje się w trybie normal, to wywoływana jest wewnętrzna funkcja _move. Na podstawie zmiennej mouse_position ustala, które kwadraty mają być zmienione,
- virtual void scramble()=0, odpowiada za mieszanie kwadratów na początku rozgrywki
- virtual void _move(sf::Vector2i mouse_position) = 0, wewnętrzna metoda, która zamienia kwadraty na podstawie miejsca kliknięcia lewego przycisku myszy,
- int get_id(const unsigned i, const unsigned j), zwraca id kwadratu na podstawie jego indeksów i,j.

Klasa Version1, dziedzicząca po klasie Board odpowiada za obsługę zdarzeń związanych z 1. wersją gry. Posiada następujące metody wirtualne:

- void scramble() override,
- void _move(sf::Vector2i mouse_position) override.

Klasa Version2 dziedziczy po klasie Board i odpowiada za poprawne działanie drugiej wersji gry. Przechowuje następujące zmienne:

- sf::Texture black_texture, tekstura przechowująca wycięty kwadrat,
- unsigned current_black_x, current_black_y, współrzędne pustego kwadratu.

Posiada następujące metody:

- void _move(sf::Vector2i mouse_position) override,
- void swap_squares(int x1, int y1, int x2, int y2), pomocnicza funkcja pozwalająca zamienić sąsiednie kwadraty o współrzędnych [x1,y1], [x2,y2].

Dodatkowo stworzono pomocnicze typy wyliczeniowe:

- dimension, przechowuje szerokość marginesów, długość boku kwadratu oraz odstępy pomiędzy sąsiednimi kwadratami
- Game_version, określa obecny stan aplikacji: czy gracz dopiero wybiera wersję gry, czy już wybrał pierwszą lub drugą wersję.

7. Testowanie

Działanie programu sprawdziliśmy, wczytując losowy jeden z przygotowanych obrazków. Po kliknięciu [1] lub [2] program losowo miesza kwadraty kolejno według 1. lub 2. wersji.

Gdy graczowi uda się poprawnie złożyć obrazek, program zgodnie z oczekiwaniami kończy działanie.

8. Wdrożenie, raport i wnioski

Podstawowe wymagania projektu zostały prawidłowo wykonane. W przyszłości można zaimplementować wymagania rozszerzone lub np. dodać kolejne sposoby mieszania kwadratów.