1. Zasady gry:

Saper (Minesweeper) to klasyczna gra logiczna, w której celem gracza jest **odkrycie wszystkich pól na planszy, które nie zawierają min**. Kliknięcie pola z miną kończy grę. Odsłonięte pola pokazują liczbę min w sąsiednich komórkach, co pozwala logicznie wywnioskować położenie pozostałych.

Gracz może:

- **Odkryć co kryje dane pole** klikając lewym przyciskiem myszy
- **Oznaczyć pewne pole z miną** klikając je prawym przyciskiem myszy – pojawi się flaga.

- **Oznaczyć pole jako niepewne**, używając ponownego kliknięcia prawym przyciskiem – pojawi się **znak zapytania (?)**, który informuje, że pole może, ale nie musi zawierać miny.

- Trzecie kliknięcie prawym przyciskiem resetuje pole do stanu neutralnego.

Znak „?” jest opcjonalną pomocą logiczną dla gracza i **nie wpływa na wynik gry** – służy jedynie jako przypomnienie. Pierwsza kliknięcie LPM na plansze jest bezpieczne i wygeneruje ono **bezpieczną przystań “Safe Zone”** czyli połączone ze sobą bezpieczne pola bez min, co pozwoli na rozpoczęcie gry i analizy pola minowego.

2. Przykładowe zastosowania współczesnych rozwiązań C++
    a. std::optional

Zapewnia nam bezpieczny sposób na deklarację zmiennych/obiektów, które mogą jeszcze nie posiadać wartości w danym momencie. Nie musimy dzięki temu przypisywać pustych wartości do zmiennych.

```
std::optional<sf::Text> title;
std::optional<sf::Text> startButton;
std::optional<sf::Text> configButton;
std::optional<sf::Text> creditsButton;
std::optional<sf::Text> exitButton;
std::optional<sf::Text> gridLabel;
std::optional<sf::Text> gameOverText;
std::optional<sf::Text> continueText;
```

    b. structured binding

Krótszy zapis dla rozpakowania danych - czytelniejszy kod

```
while (!q.empty() && safeZone.size() < count) {
    auto [x, y] = q.front(); q.pop();
```

```
// 2. Also forbid placing mines around the safe zone
std::unordered_set<std::pair<unsigned int, unsigned int>, pair_hash> forbidden;
for (const auto& [x, y] : safeZone) {
```