

# SOKOBAN

Magdalena Małczyk

## 1. O grze Sokoban

Projektem jest japońska łamigłówka Sokoban. Gra jest oparta na idei magazynu z pudłami, które należy przesunąć w wyznaczone miejsca.

Link do Sokoban Wiki:

[http://sokobano.de/wiki/index.php?title=Main\\_Page](http://sokobano.de/wiki/index.php?title=Main_Page)

## 2. Moja wersja gry

Gra będzie dostępna w dwóch wersjach:

- z poziomami generowanymi automatycznie,
- z poziomami ładowanymi z podanego pliku o specyficznym formacie

Po uruchomieniu gry włącza się ekran startowy z wyświetlonymi instrukcjami. Z ekranu startowego można przejść do trybu **single player** albo **two player**. Wyświetlana jest odpowiednio jedna lub dwie plansze. Przepchnięcie wszystkich gwiazdek na wyznaczone miejsca powoduje pojawienie się komunikatu „you won”.

Błędy w poziomie, który próbuje wyświetlić gra, wywołują ekran błędu.

Opis przyjętego formatu pliku poziomów:

[http://www.sokobano.de/wiki/index.php?title=Level\\_format](http://www.sokobano.de/wiki/index.php?title=Level_format))

## 3. Sterowanie postacią

W ekranie startowym:

- S** – przejście w tryb single player
- D** – przejście w tryb two player

W czasie gry:

Klawisze **strzałek** – sterowanie w *single player* i sterowanie postacią na prawej planszy w *two player*.

Klawisze **WSAD** – sterowanie postacią na lewej planszy w *two player*.

**Backspace** – reset planszy do stanu początkowego.

**Tab** – przejście do ekranu start

**N** – przejście do następnego poziomu

**P** – przejście do poprzedniego poziomu

Po wygranej grze dowolny klawisz przełącza na następny poziom.

#### 4. Architektura projektu

Użyty został Python 2.7 w wydaniu Anaconda. Zainstalowano dodatkowo moduł Pygame.

<http://www.pygame.org/wiki/about>

Projekt jest podzielony na 5 pakietów:

##### **levels:**

- klasa **LevelReader**, która zawiera metody pozwalające na odczytanie pliku poziomów. Tworzy listę odpowiednio przygotowanych par obiektów typu **Board**, omijając błędy w pliku.
- klasa **Board**, która umożliwia wykonywanie na planszy operacji takich jak reset, krok, lustrzane odbicie.
- klasa **Map**, która jest bezpośrednią reprezentacją planszy w programie. Edytuje otrzymaną dwuwymiarową tablicę w celu zebrania niezbędnych danych i ozdobienia planszy.
- klasa **Generator**, (nie zaimplementowana), która automatycznie generuje poziomy zamiast je odczytywać.

##### **players:**

- klasa **Player**, której funkcja *command* wybiera polecenie do przekazania klasie **Window** na podstawie zdarzenia z klawiatury i jego kontekstu.

##### **visualizers:**

- klasa **Graphics**, która wczytuje wszystkie pliki używane w grafice gry i segreguje je w słowniki.
- klasa **GameWindow**, która odpowiada za wszystko co widzi gracz, a także przechwytyje zdarzenia z klawiatury do przekazania klasie **Player**.

##### **game\_engine:**

- klasa **PlainGameEngine**, która steruje komunikacją klas **Player** i **Window** oraz przełącza poziomy do wyświetlenia. Używa klasy **LevelReader**.
- klasa **AdvancedGameEngine**, (nie zaimplementowana); jak wyżej, z tym że używa klasy **Generator**.

Tutorial użyty przy nauce tworzenia grafiki i logiki gier komputerowych:

„*Making Games With Python & Pygame*”, Albert Sweigart.

