Lazy Snek

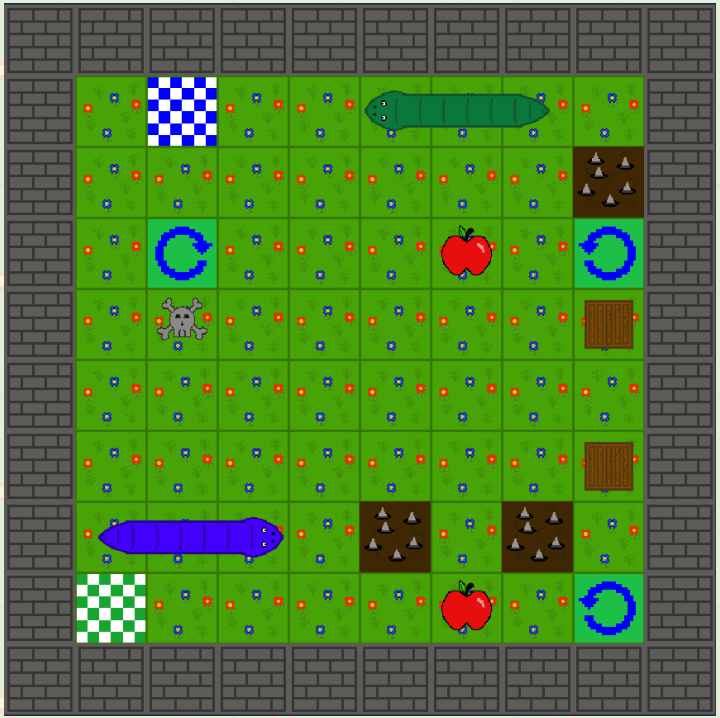
Desktopowa gra logiczna tworzona w ramach projektu zespołowego na przedmiocie Inżynieria Oprogramowania, 2020/21

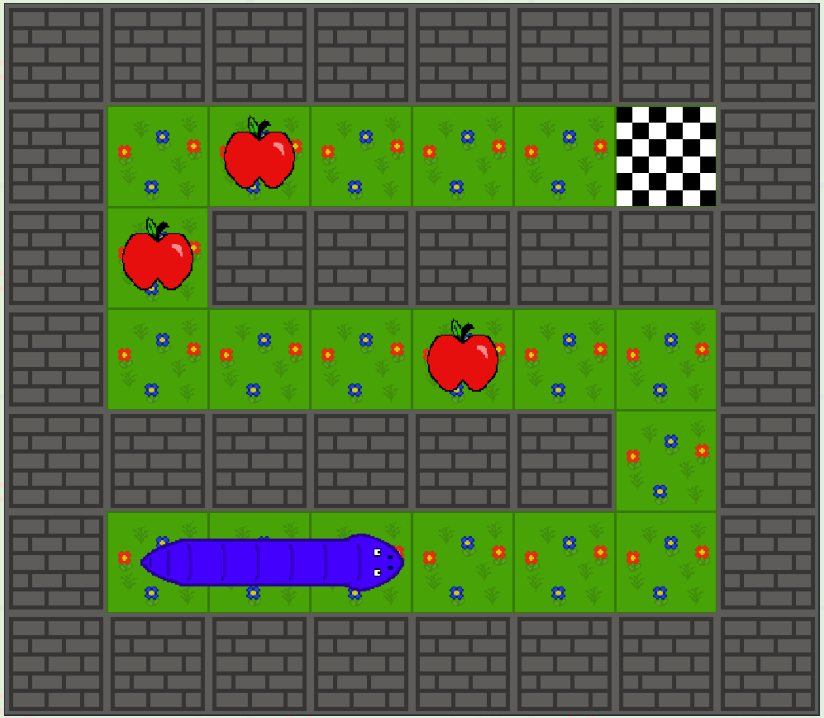
Kamil Jonak, Władysław Pałucki, Maciej Sygnowski, Konrad Załęcki

Zasady gry

- Rozwiązywanie poziomów gry polega doprowadzeniu węży znajdujących się na dwuwymiarowej planszy do punktów docelowych.
- Oprócz węży na planszy znajdują się różne bloki, które mogą wchodzić w interakcję z wężami i innymi blokami.

Zasady gry

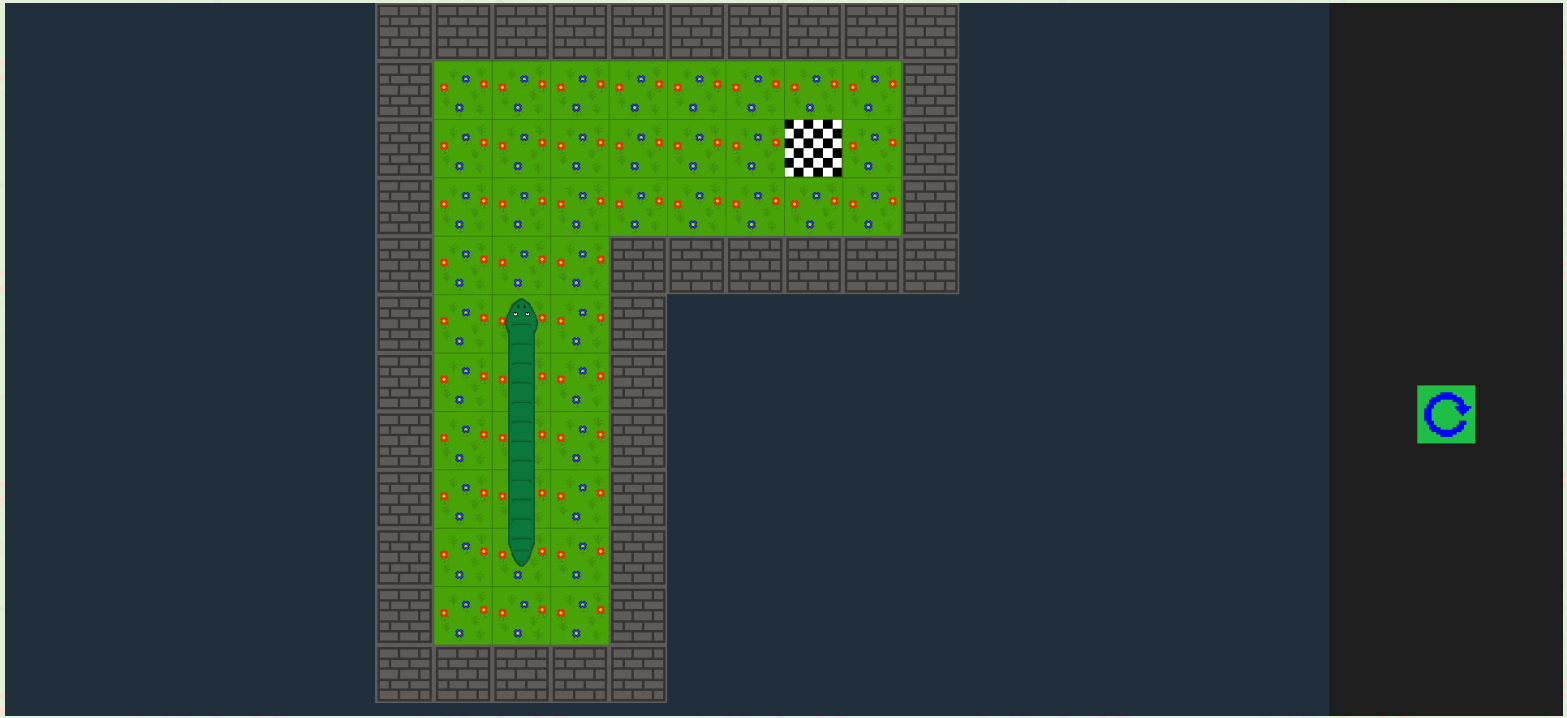




Zasady gry

- Rozwiązywanie poziomów polega na postawieniu na planszy w odpowiedni sposób dodatkowych bloków, które gracz ma do dyspozycji w każdym poziomie.

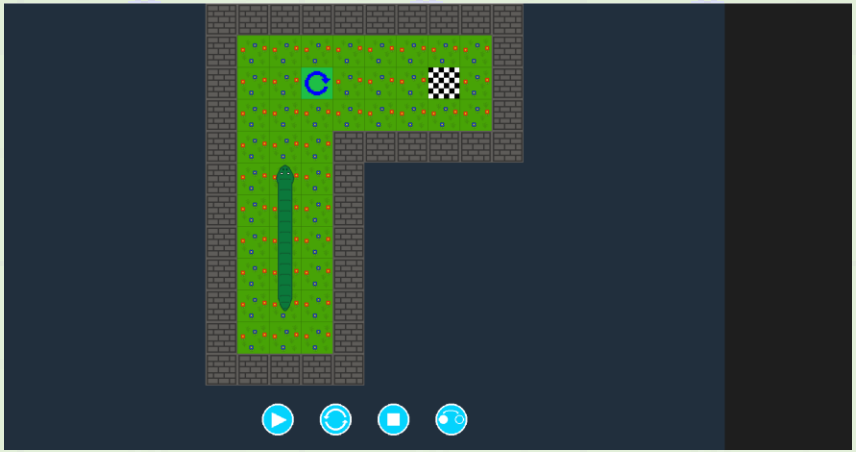
Zasady gry

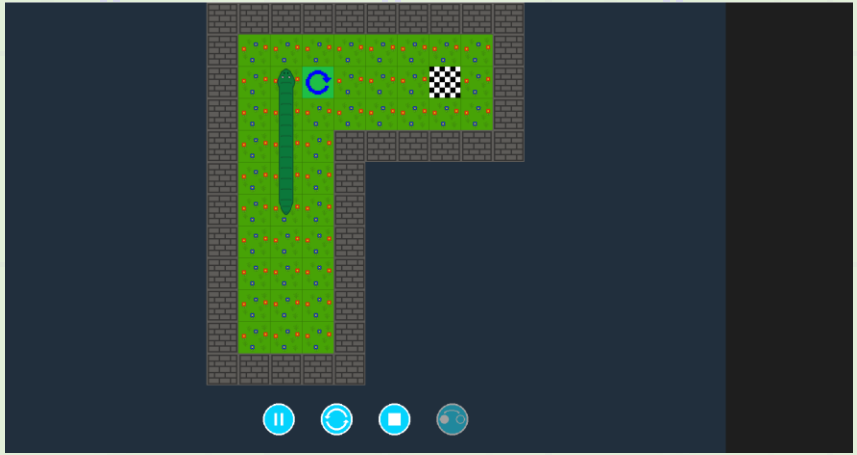


Zasady gry

- W dowolnym momencie gracz może przeprowadzić symulację rozwiązywanego poziomu.
- W symulacji węże znajdujące się na planszy przesuwają się co jednostkę czasu o jedno pole do przodu.

Zasady gry

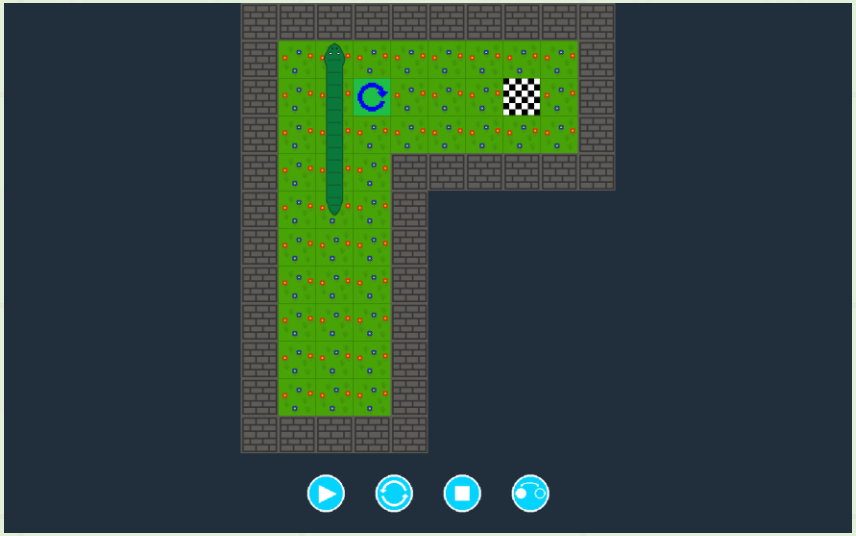


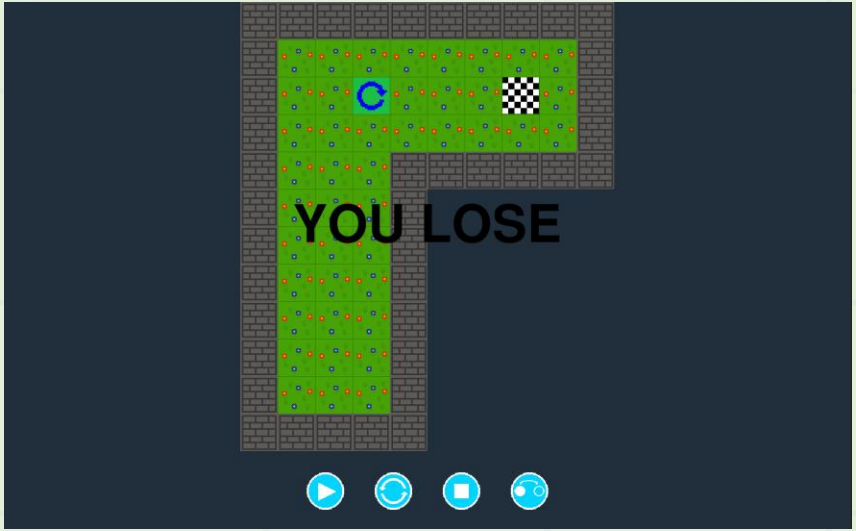


Zasady gry

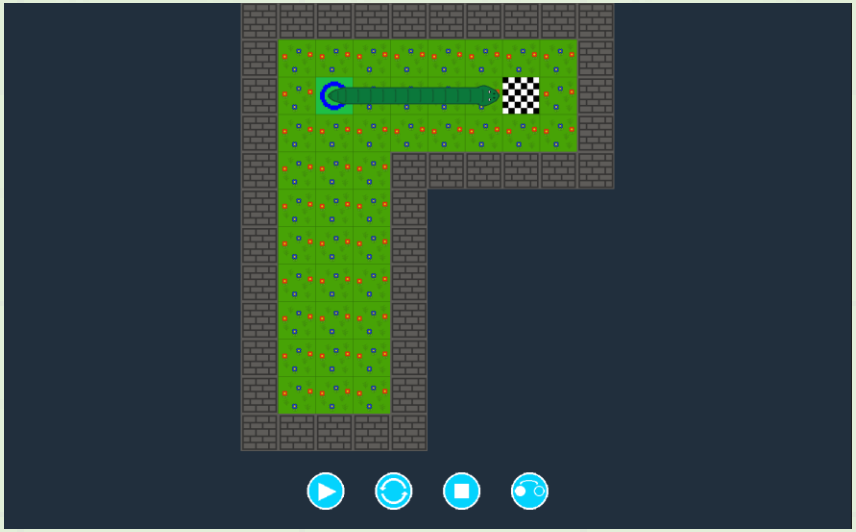
- Poziom zostaje wygrany, gdy w wyniku symulacji wszystkie węże dotarły do mety.
- Jeśli podczas symulacji zginie dowolny wąż, wówczas próba przejścia poziomu zakończyła się porażką i trzeba spróbować ponownie.

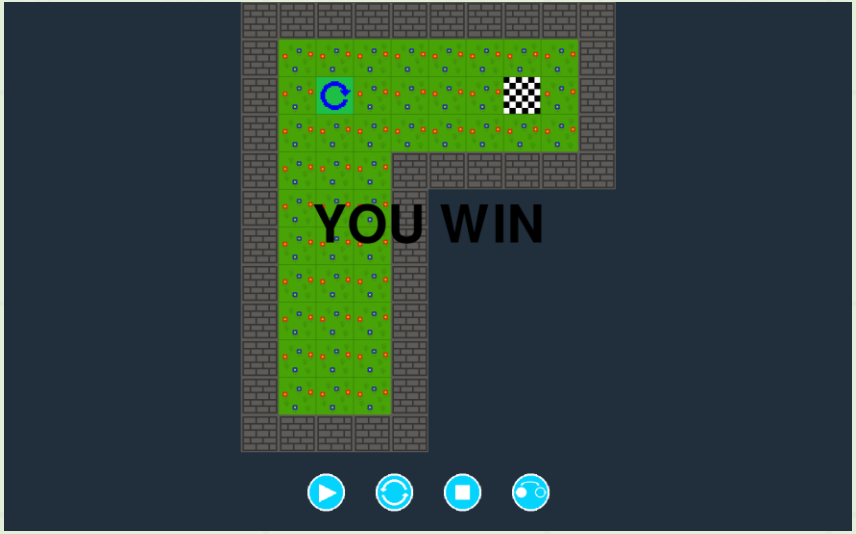
Zasady gry





Zasady gry





Silnik gry

- Głównymi komponentami silnika gry są klasy reprezentujące:
  - planszę
  - pole planszy
  - blok
  - węża
  - poziom

Silnik gry

- Plansza składa się z pól, na których mogą znajdować się bloki i węże.
- Obecnym stan poziomu jest przechowywany w obiekcie klasy reprezentującej poziom.
  Ta sama klasa przeprowadza symulację poziomu.

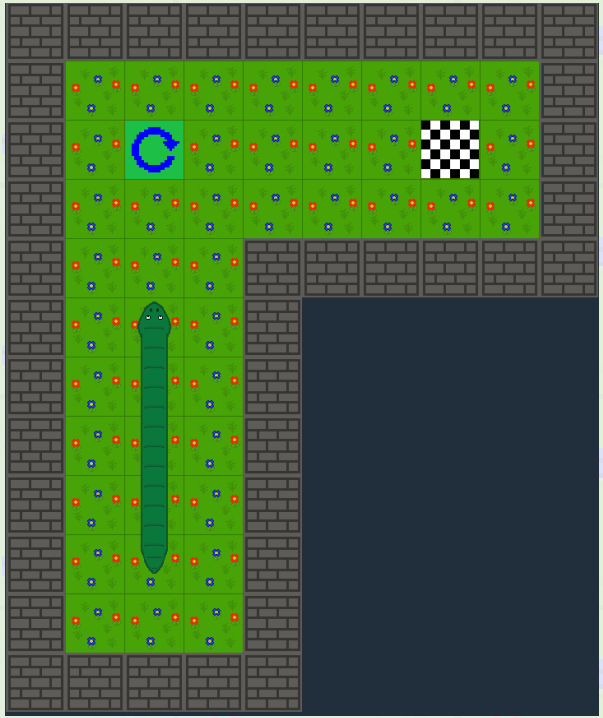
Silnik gry - bloki

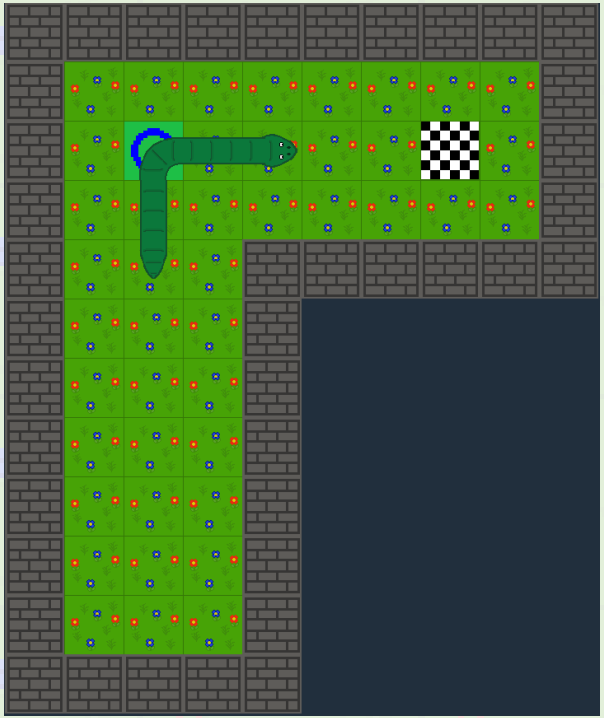
- Bloki dzielą się na 2 główne typy:
  - *Convex*
  - *Flat*
- Główną różnicą pomiędzy tymi typami jest to, że bloki Convex mogą są przesuwalne, natomiast bloki flat nie mogą zmienić swojego położenia podczas symulacji.

Silnik gry – bloki typu *Flat*

- Nieprzesuwalne i niezniszczalne
- Jeśli wejdzie na nie jakiś wąż, mogą wchodzić z nim w interakcję, zmieniając jego stan, np. kierunek poruszania się
- Bloki typu *Convex* mogą wepchnięte i zepchnięte z bloku typu *Flat*, tak samo jak z pustego pola.

Silnik gry – przykład bloku typu *Flat*

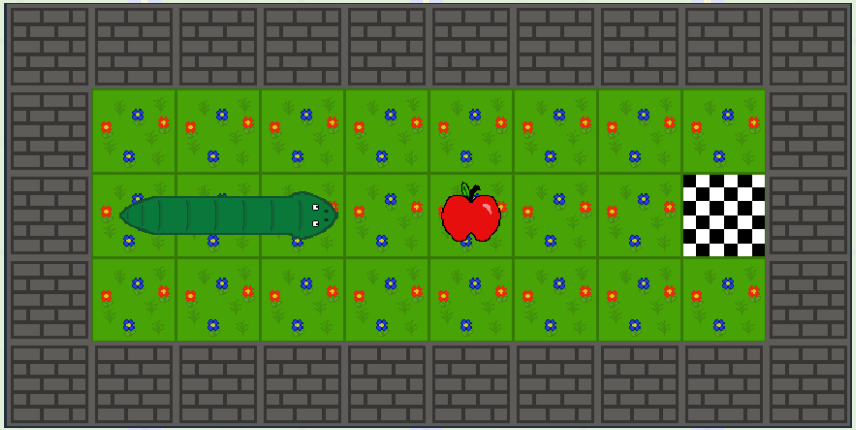


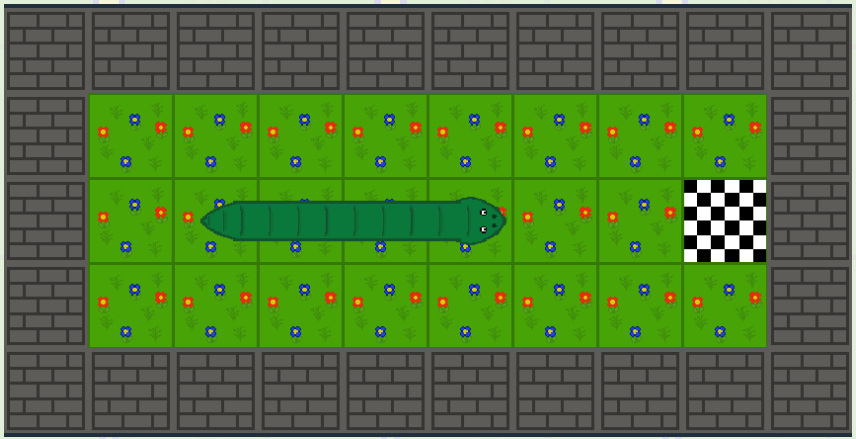


Silnik gry – bloki typu *Convex*

- przesuwane przez inne bloki typu *Convex*
- mogą wchodzić w interakcję z wężem, który próbuje na nie wjechać, zmieniając stan węża
- mogą zostać zniszczone w wyniku interakcji

Silnik gry – przykład bloku typu *Convex*





Silnik gry - pole

- Każde pole planszy składa się z 3 warstw:

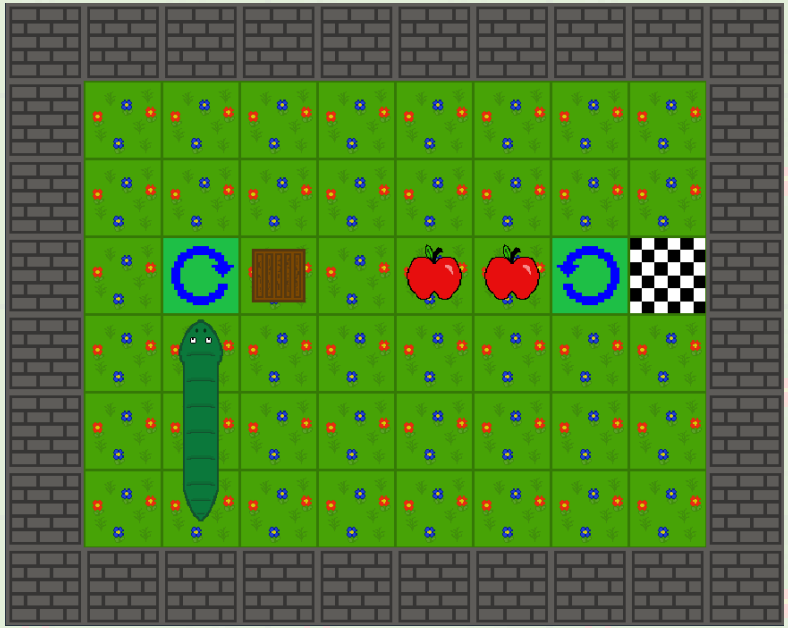
warstwa	co zawiera
Convex	Blok typu Convex lub fragment węża
Flat	Blok typu Flat
warstwa pola	Puste pole

Silnik gry - pole

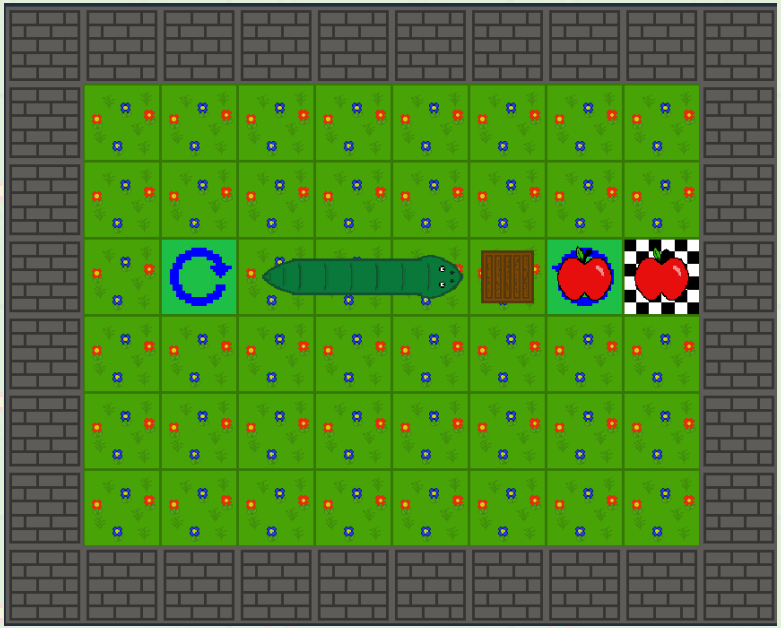
- Podczas rozwiązywania poziomu na każdym polu może znaleźć się maksymalnie jeden blok, niezależnie od tego, jakiego jest typu.
- W czasie symulacji blok typu *Convex* może zostać wepchnięty na pole, na którym znajduje się blok typu *Flat*.

Silnik gry - pole

Podczas rozwiązywania poziomu na każdym polu może znajdować się co najwyżej 1 blok lub segment węża



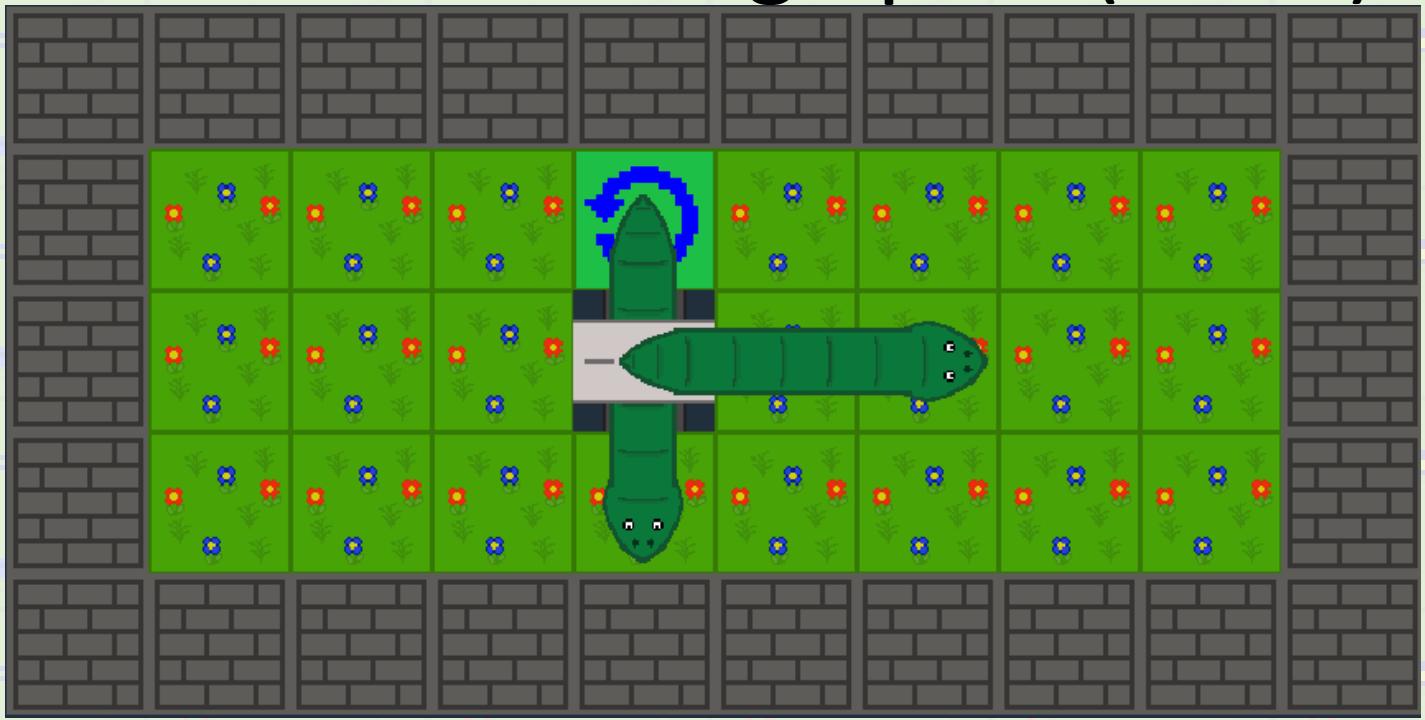
Podczas symulacji rozwiązanego poziomu bloki typu *Convex* i węże mogą znaleźć się na blokach typu *Flat*



Silnik gry - pole

- Niektóre mechaniki są implementowane przez niestandardowe typy pól
- Te typy pól komunikują się z blokami i wężami w ten sam sposób, co normalne pola, ale zachowują się inaczej

Silnik gry – przykład niestandardowego pola (tunel)



Poziomy w grze

Poziomy dostępne w aplikacji dzielą się na 2 grupy:

- bazowe – pula poziomów udostępniona razem z grą
- niestandardowe – tworzone przez użytkowników, które można zaimportować do gry z pliku

Edytor poziomów

Gra udostępnia również edytor poziomów, w którym użytkownicy mogą stworzyć własne poziomy i wyeksportować je do pliku. Następnie można dodać je do puli niestandardowych poziomów.

Użyte narzędzia i technologie

Zarządzanie projektem zostało w całości zrealizowane za pomocą funkcji udostępnianych w serwisie Github

- Repozytorium kodu na Github
- CI skonfigurowane przy użyciu Github Actions
- Backlog i scrum board przy użyciu projektu na Github
- Dokumentacja kodu i inne dokumenty w wiki na Github

Użyte narzędzia i technologie

Aplikacja została napisana w całości przy użyciu języka Python (w wersji 3.8). Przy wyświetlaniu zawartości oraz komunikacji z użytkownikiem użyto biblioteki *pygame*, w nielicznych przypadkach wspomagając się również biblioteką *Tkinter*.

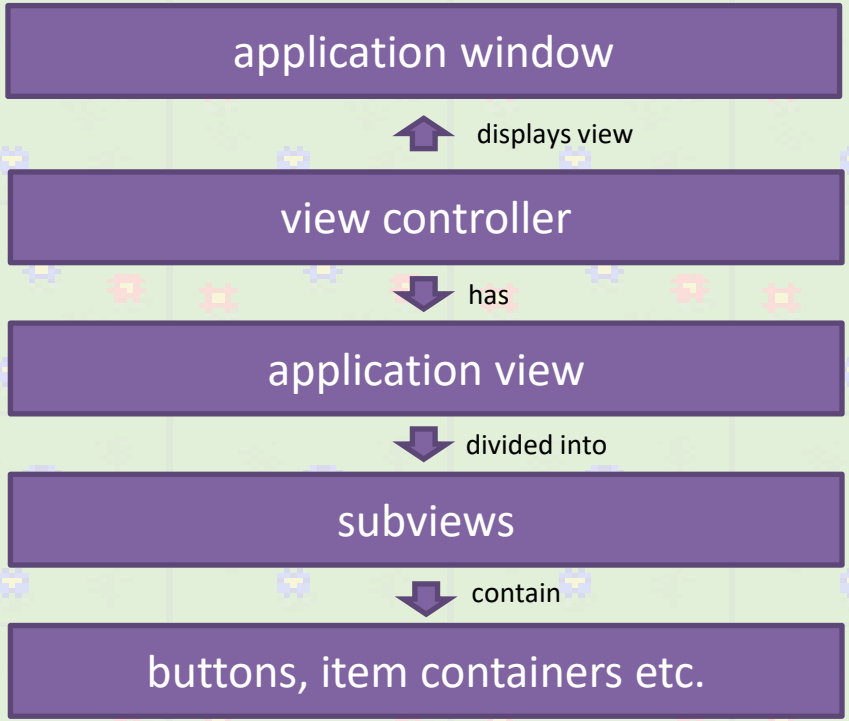
Testy do projektu zostały napisane z użyciem biblioteki *pytest*. Zgodność stylu kodu z PEP 8 była sprawdzana przy użyciu modułu pythona *flake8*.

Wszystkie grafiki użyte w projekcie zostały stworzone w edytorze dostępnym w serwisie *piskelapp.com*.

Aplikacja – wyświetlanie zawartości

- Tworzenie okna aplikacji oraz wyświetlanie tekstur w jego odpowiednich miejscach są realizowane przy pomocy biblioteki *pygame*
- Tworzenie wyświetlanej zawartości i zarządzanie nią jest realizowane przy pomocy wzorca *Model-View-Controller*

Aplikacja – wyświetlanie zawartości



Aplikacja – komunikacja z użytkownikiem

- Realizowana również przy pomocy biblioteki *pygame*, z użyciem typu *pygame.event*
- Wyświetlane elementy obsługują akcję danego typu lub przekazują ich obsługę swoim komponentom
- W wyniku obsługi każdej akcji stan wyświetlanego widoku może się zmienić lub może zostać utworzony nowy widok do wyświetlenia

Aplikacja – komunikacja z użytkownikiem