**Gra w Statki**

**Dokumentacja dla dewelopera**

Całość gry zawiera się w module **battleships**. Znajdują się tam trzy główne foldery:

* **Data** – przechowuje wszystkie pliki potrzebne do działania gry. Są to obrazki przycisków i obrazek tytułowy.
* **Tests** – folder z testami.
* **Src** – folder ze wszystkimi skryptami. Jest w nim także folder **Classes**, który zawiera trzy klasy.

Do stworzenia gry zostały wykorzystane dwa główne moduły: **pygame** oraz **numpy**. Pierwszy odpowiada za interfejs graficzny gry. Dzięki niemu można rysować planszę i przyciski, a także wykrywać różne sygnały wejścia ze strony gracza. Z kolei na module **Numpy** oparta jest cała logika gry. Każda plansza ma swoje odzwierciedlenie w postaci odpowiedniej macierzy. Dzięki temu można łatwo i szybko uzyskiwać informacje o stanie poszczególnych pól planszy.

Każde pole planszy opisane jest koordynatami x i y które odpowiadają koordynatom pól w macierzy. Każde z pól jest też opisane cyfrą symbolizującą stan pola. Te stany to:

* 0 – puste pole,
* 1 – pole na którym znajduje się komórka statku,
* 2 – pole na którym znajduje się zniszczona komórka statku,
* 3 – puste, ale trafione pole.

Za każdym razem gdy dokonuje się jakakolwiek zmiana na planszy, macierz musi być odpowiednio aktualizowania. Inaczej dojdzie do sytuacji w której choć na planszy pole będzie wydawało się zapełnione, to w rzeczywistości będzie traktowane przez grę jako puste.

Gra jest podzielona na trzy główne ekrany: ekran tytułowy, ekran do rozmieszczenia statków na planszy oraz ekran na którym odbywa się właściwa rozgrywka. Każdy z ekranów posiada funkcję main, która odpowiada za zarządzanie wszystkimi sygnałami wejścia. Gdy nastąpi odpowiedni warunek, wywoływana jest funkcja kolejnego ekranu i gra przechodzi do kolejnego stanu.

Każdy ekran posiada swój własny plik główny, wraz z funkcjami które są mu potrzebne. Ale oprócz tego istnieją także funkcje, które znajdują się w innych, wyodrębnionych plikach. Te pliki to:

* **constants.py** – znajdują się w nim wszystkie stałe potrzebne używane przez większość plików gry. Dzięki takiemu rozwiązaniu wystarczy zmienić jedną linijkę kodu aby zmienić np. kolor tła.
* **check\_placement.py** – znajdują się w nim trzy funkcje odpowiedzialne za sprawdzenie, czy można położyć statek w danym miejscu na planszy. Sprawdzane jest czy statek nie znalazłby się poza planszą, czy nie nachodziłby na inne statki i czy nie byłby za blisko innych statków. Te funkcje są też potem używanie przy losowym wygenerowaniu planszy dla gracza komputerowego.
* **draw\_objects.py** – znajdują się w nim trzy funkcje. Pierwsza rysuje statek odpowiedniego typu. Każdy statek w grze ma przypisany swój kod. Liczby od 1 do 4 symbolizują statki poziome – 1 to pojedynczy kwadrat, 2 to dwa kwadraty itd. Liczby od 5 do 8 symbolizują statki pionowe. Jak można zauważyć, dla pojedynczego kwadratu występują dwie reprezentacje – 1 i 5. W ten sposób zostaje zachowana symetria i prościej jest zarządzać resztą kodu. Druga rysuje menu drugiego ekranu. Trzecia obrysowuje statek niebieską obwódką gdy wszystkie jego części zostaną zniszczone.
* **create\_objects.py** – te funkcje tworzą wszystkie obiekty potrzebne dla drugiego ekranu. Ich pozycje na ekranie są nawzajem od siebie uzależnione. Dzięki temu przesunięcie np. planszy przesunie także tekst i przyciski.
* **array\_methods.py** – znajdują się tam funkcje odpowiedzialne za manipulowaniem macierzami i sprawdzaniem ich stanu.
* **enemy\_moves.py** – w tym pliku znajduje się cała logika odpowiedzialna za generowanie ruchów dla gracza komputerowego.

Gra posiada także trzy główne klasy:

* **Board** – klasa odpowiedzialna za wyprodukowanie planszy. Plansza posiada własną przestrzeń dzięki której gra wie, na czym ma rysować, a także swój rozmiar i swoje położenie na głównym ekranie. Nie posiada za to informacji o statkach – te umieszczone są w macierzy.
* **Button** – ogólna klasa dla przycisków. Każdy przycisk posiada dwa obrazki – jeden gdy nie jest kliknięty, i drugi gdy zostanie kliknięty. Po każdej zmianie stanu przycisku należy wywołać funkcję update, aby obrazek przycisku został odpowiednio zmieniony. Ta klasa posiada trzy podklasy: **SelectionButton**, czyli przyciski odpowiedzialne za zmianę typu statku, **RotateButton**, czyli przycisk rotacji który posiada także funkcję służącą do zmiany wartości zmiennej choice, oraz **SubmitButton** – specjalny przycisk bez drugiego obrazka który posiada także parametr opisujący czy można go kliknąć.
* **ShipsText** – klasa odpowiedzialna za monitorowanie ilości statków pozostałych do położenia. Każdy obiekt tej klasy jest przypisany do innego typu statku, a więc też do odrębnego przycisku.

**Ekran rozstawiania statków**

Funkcja **main** w pierwszej kolejności tworzy wszystkie potrzebne sobie obiekty. Następnie tworzona jest główna pętla gry. Jest to pętla typu while True. W każdej iteracji pętli rysowane są wcześniej stworzone obiekty. Następnie sprawdzane jest, czy gracz położył już wszystkie swoje statki. Jeśli tak, to pojawia się przycisk SUBMIT. Tak naprawdę ten przycisk cały czas tam jest. Jedyne co się zmienia po spełnieniu odpowiednich warunków, to jego stan – staje się klikalny.

Następnie gra nasłuchuje sygnałów od gracza. Jeśli zostanie wciśnięty lewy przycisk myszy, to gra sprawdza co zostało naciśnięte.

* Jeżeli to był któryś z przycisków wyboru statku, to następuje zmiana aktualnego na wybrany i zmiana wartości zmiennej *choice*.
* Jeżeli był to przycisk SUBMIT, to gra przechodzi do kolejnego ekranu.
* Jeżeli był to przycisk rotacji, to gra zmienia aktualną wartość zmiennej choice oraz zmienia stan przycisku rotacji. Jest to potem wykorzystywane przy wyborze typu statku.
* Jeżeli kliknięcie nie było kliknięciem na przycisk, to gra sprawdza czy gracz nie wybrał pola na planszy. Aby to zrobić, gra przez cały czas monitoruje pozycję myszki, a następnie przekłada ją na konkretne koordynaty planszy. Jeżeli kliknięcie nastąpiło w jej obrębie, to gra sprawdza czy w danym miejscu można położyć statek określonego typu. To sprawdzenie odbywa się za pomocą funkcji w pliku **check\_placement.py**. Jeśli położenie się powiedzie, to gra rysuje odpowiedni statek na planszy, a następnie aktualizuje macierz i tekst obok odpowiedniego przycisku.

Jeżeli zostanie wciśnięty prawy przycisk myszy, to gra sprawdza jedynie czy kliknięcie nastąpiło w obrębie planszy i czy było to kliknięcie na statek. Jeśli tak, to statek jest usuwany, a macierz i tekst są aktualizowane.

Usuwanie statku polega na analizie pól dookoła pola na które kliknął gracz. Gra najpierw określa oś statku, a następnie sprawdza wszystkie pola w danym kierunku aż nie dotrze do pustego pola. Wtedy funkcja zaczyna sprawdzać pola w przeciwnym kierunku, a gdy dotrze do pustego pola, kończy swoje działanie.

**Ekran głównej rozgrywki**

Funkcja main najpierw tworzy dwie plansze: planszę gracza i planszę przeciwnika. Do planszy gracza jest dołączona macierz z poprzedniego ekranu. Na jej podstawie gra rysuje wszystkie położone wcześniej statki. W przypadku gracza komputerowego gra dokonuje losowego rozłożenia statków. Dla każdego statku dokonują się trzy losowania – dwa na koordynaty x i y, oraz trzecie, na zdecydowanie czy statek będzie ułożony pionowo czy też poziomo. Następnie za pomocą funkcji w pliku **check\_placement.py** sprawdzane jest, czy to wylosowanie jest słuszne. Jeśli tak, to gra losuje kolejny statek. Dzieje się to do momentu aż nie zostaną wylosowane wszystkie potrzebne statki.

Po narysowaniu plansz rozpoczyna się główna pętla. W niej sprawdzane są dwie rzeczy: czy gracz kliknął na swoją planszę i czy któryś z graczy nie wygrał już rozgrywki. W przypadku kliknięcia następuje analiza klikniętego pola.

Jeśli to pole było puste, to gra ustawia odpowiedni stan w macierzy i koloruje kwadrat na niebiesko. Jeśli to pole zawierało statek, to gra koloruje ten statek na czerwono, a następnie sprawdza czy kliknięty kwadrat nie był ostatnią częścią statku. Jeżeli tak, to zostaje wtedy wywołana funkcja odpowiedzialna za otoczenie statku niebieskimi kwadratami tak, żeby nie można było oddać już tam strzału. Oczywiście macierz gracza jest cały czas po drodze aktualizowana.

Po ruchu gracza następuje ruch gracza komputerowego. Jego zachowanie można podzielić na dwa różne stany. W pierwszym stanie kolejne punkty do strzału są kompletnie losowe. Kolejne strzały następują do momentu aż komputer nie trafi w statek. Wtedy komputer zmienia swoje zachowanie. Na podstawie obecnego stanu planszy i danego punktu generowany jest zestaw instrukcji dla komputera.

Cała generacja przebiega w pliku **enemy\_moves.py**. Tą generację także można podzielić na dwa etapy. Pierwszy etap to określenie kierunku statku – poziomego lub pionowego. Komputer ma 25% szans na strzelenie w jednym z czterech kierunków: góra, dół, lewo i prawo. Po każdym strzale koordynaty strzelonego pola są dopisywane do listy poleceń. Jeśli ten strzał był pudłem, to losowanie powtarza się. Kolejne losowania przebiegają tak długo, aż:

* Komputer nie trafi w statek – przechodzi wtedy do kolejnego etapu
* Komputer trafi w każde z czterech pól i spudłuje – wówczas zwracana jest pusta lista, gdyż to oznacza że statek składa się w pojedynczego kwadratu.

Kolejnym krokiem generacji jest poruszanie się wzdłuż linii statku. Komputer ma 50% szans na udanie się w jednym kierunku i 50% szans na udanie się w przeciwnym kierunku. Komputer będzie poruszał się w danym kierunku tak długo, aż nie napotka na puste pole. Wtedy to zacznie iść w przeciwnym kierunku. Gdy napotka na puste pole po raz drugi, to będzie wtedy oznaczało że objął już swoimi ruchami cały statek.

Ostatnim krokiem jest usunięcie duplikatów z listy poleceń, oraz usunięcie pola które zostało podane jako argument wywołania funkcji. Tak spreparowana lista poleceń jest zwracana komputerowi, który następnie zamiast losowo wybierać kolejne pola kieruje się właśnie tą listą. Gdy długość listy dobiegnie do zera, komputer wraca do pierwotnego stanu i ponownie zaczyna losować punkty do strzału.

Należy zauważyć, że podczas generowania listy poleceń nie następują żadne zmiany ani na planszy, ani w macierzy. Zmiany następują dopiero gdy gra wybierze pole z listy poleceń i dokona strzału.

Gra toczy się w ten sposób dopóki któremuś z graczy nie skończą się statki. Wtedy to gra wyświetla odpowiedni komunikat i „więzi” gracza w nowej pętli. W ten sposób gra nie wykrywa już kliknięć na planszę, a jedynie wciśnięcie przycisku ENTER.

**Czego nie zdążyłem dodać**

* Jednym z pomysłów było wprowadzenie różnych poziomów trudności. Podczas tworzenia sztucznej inteligencji dla gracza komputerowego, przez przypadek stworzyłem taki algorytm, którego niemal nie dało się pokonać.
* Muzyki i efektów dźwiękowych
* Powrotu do ekranu startowego po zakończeniu rozgrywki.

**Podsumowanie**

Choć po drodze napotkałem wiele trudności i frustracji, to finalnie miło pracowało mi się nad tym projektem. Gdybym miał więcej czasu, to na pewno byłbym w stanie jeszcze bardziej doszlifować ten projekt i dodać jeszcze więcej funkcji. Natomiast już w tym stanie gra jest jak najbardziej grywalna.