Układanka wykorzystująca zdjęcia z galerii użytkownika

Kamil Stanisławski

Projektowanie aplikacji mobilnych

Celem zadania jest aplikacja umożliwiająca użytkownikowi zrelaksowanie się np. podczas siedzenia w autobusie czy czekania w kolejkach. Użytkownik wybiera obraz i poziom trudności (Łatwy, Średni, Trudny), po czym algorytm miesza fragmenty obrazu (kafelki). Zadaniem gracza jest ułożenie kafelek w spójny obraz. Gracz może przesuwać kafelki tylko na puste pole. Gdy obraz jest spójny gra się kończy i użytkownik otrzymuje informacje o ilości ruchów które wykonał. Trudności różnią się ilością przesunięć kafelek przez algorytm.

Wymagania funkcjonalne:

- Wybór trudności
- Ładowanie obrazka z pamięci telefonu
- Algorytmiczne przetasowywanie kafelek obrazka
- Układanie obrazka za pomocą przesuwania palca po ekranie dotykowym
- Po ułożeniu obrazka, wyświetlanie ilości ruchów
- Po ułożeniu obrazka, możliwość rozpoczęcia od nowa lub zmiany poziomu trudności

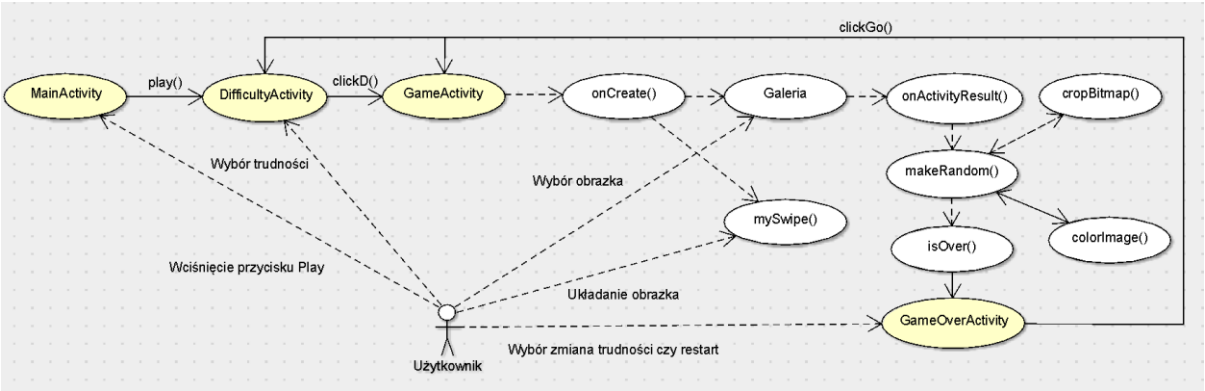
Wymagania niefunkcjonalne:

- Tworzymy aplikacje na android 8.1
- Takie samo tło dla wszystkich aktywności, oprócz tła GameActivity
- Taki sam kolor guzików

Instrukcja obsługi:

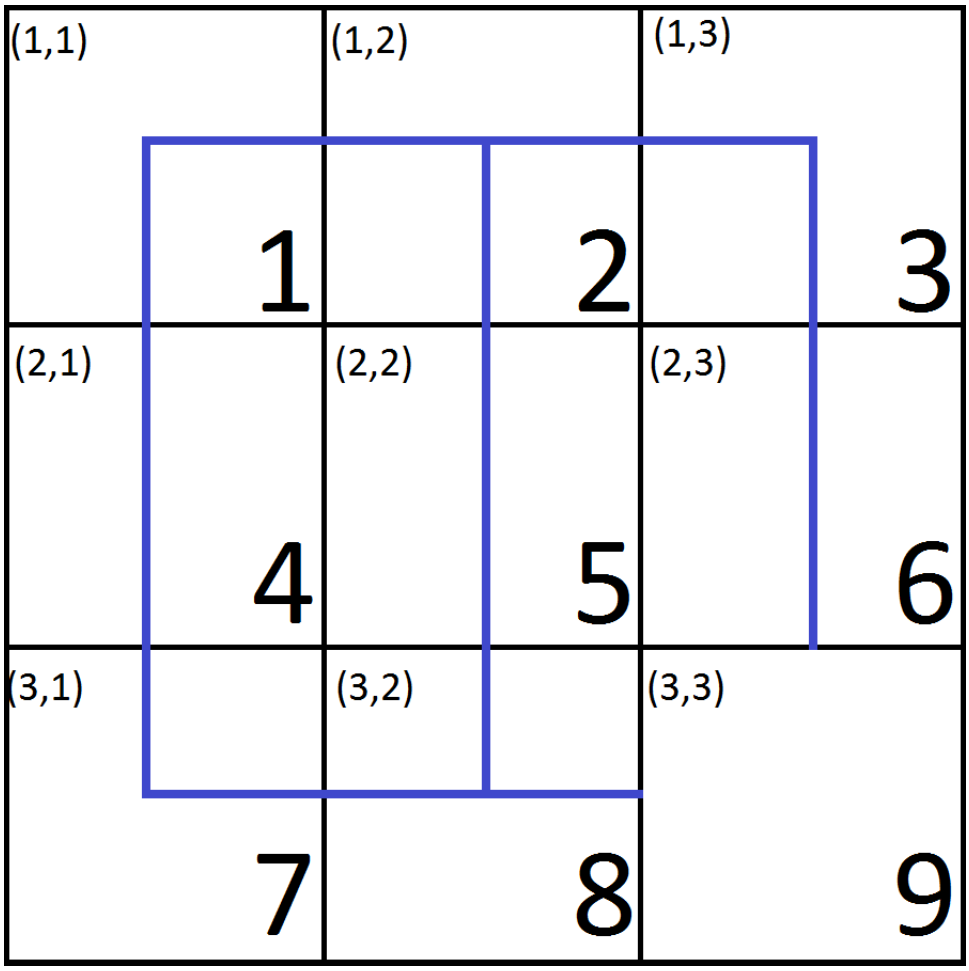
1. Wybieramy poziom trudności
2. Wybieramy obrazek, który chcemy ułożyć
3. Układamy obrazek przesuwając kafelki na puste pole, za pomocą ruchu przesuwania palca po ekranie
4. Po ułożeniu obrazka możemy ponownie zmienić trudność rozgrywki lub zresetować grę i zacząć jeszcze raz
5. Grę używamy wyłącznie w trybie pionowym - wertykalnym

Diagram użycia:



Opis klas i metod:

- class Title – Klasa ta zawiera położenie, bitmapę oraz numer bitmapy. Jest ona częścią programistyczną gry właściwej, czyli w jej obiektach przechowujemy części bitmapy wraz z ich położeniem na “planszy”. Gdy numer bitmapy zgadza się z położeniem wiemy, że użytkownik ułożył spójny obraz.



- class OnSwipeTouchListener – Klasa ta zajmuje się sprawdzeniem w którą stronę użytkownik przesunął palec po ekranie. Nadpisuje on klasę OnTouchListener. Używamy jej instancji w metodzie mySwipe().

- class GameActivity – Tworzymy w niej wiele zmiennych takich jak.
    - diff która odpowiada za trudność rozgrywki. Jest zmienną globalną co pozwala nam na jej zmianę w klasie DifficultyActvity.
    - score która jest ostatecznym wynikiem naszej rozgrywki. Jest ona zwiększana w metodzie DrawView() i resetowana w metodzie onCreate().
    - ImageViewX która jest reprezentacją obrazka na ekranie użytkownika.

  Tworzymy również obiekty klasy Title.

- onCreate() w klasie GameActivity – Metoda ta zajmuje się:
    - Zerowaniem zmiennej score.
    - Tworzeniem i nadawaniem wartości obiektu size, rozdzielczości ekranu.
    - Dodaniem obiektów klasy Title utworzonych w klasie GameActivity do listy arrayList.
    - Łączeniem zmiennej przechowującej bitmapy imageViewX z odpowiadającym mu polem ImageView wyświetlającym bitmapę.
    - Wyświetlaniem galerii by użytkownik mógł wybrać obrazek.
    - Wywołuje metodę mySwipe().

- onActivityResult() - Wczytuje obrazek z galerii, skaluje go do szerokości ekranu i przekazuje go w metodzie makeRandom().

- makeRandom() - Metoda ta najpierw miesza ustawienie “płytek obrazu” zgodnie z zasadami gry, a następnie przyporządkowuje je odpowiednim obiektom klasy Title. Dzieląc przekazaną bitmapę na odpowiednie części. Część dziewiąta otrzymuje kolor przeźroczysty i zostaje ona “pustym” kafelkiem. Na koniec wywołuje metodę DrawView().

- cropBitmap() - Wycina z przekazanej bitmapy odpowiedni fragment.

- colorImage() - Tworzy bitmapę o określonych rozmiarach i kolorze.

- DrawView() - Nadaje polom wyświetlającym obrazy, odpowiednią bitmapę z obiektów klasy Title. Dodaje jeden do klasy score. Wywołuje metodę isOver();

- isOver() - Metoda ta sprawdza, czy obraz został poprawnie ułożony. Jeśli tak, przechodzimy na aktywność GameOverActivity.

- mySwipe() - Metoda ta sprawdza, czy użytkownik przesunął palcem po ekranie i jeśli tak to w jaką stronę. Następnie zamienia “pusty” kafelek z kafelkiem wybranym przesunięciem przez użytkownika i wywołuje metodę DrawView().

Wnioski:

Algorytm odpowiedzialny za mieszanie kafelek nie może losowo zmieniać kolejności fragmentów, bo wtedy użytkownik nie będzie mógł ułożyć obrazka. Jedynym sposobem by to zrobić jest pomieszanie kafelek tak jakby zrobił to sam użytkownik, czyli według zasad gry. Ale takie podejście wymaga bardzo dużej ilości iteracji z powodu zasad jakie to podejście narzuca. Jako że mamy do wyboru cztery ruchy kafelkiem z czego niektóre z nich mogą być niemożliwe w aktualnej pozycji kafelka, niektóre iteracje przepadają. Kolejnym problemem jest powrót kafelka na

poprzednie miejsce co może wydawać się dziwne w końcu mamy w teorii 25% szans na powrót. Ale jeśli spojrzymy na pozycje startową kafelka która umożliwia tylko dwa kierunki, a następnie tylko trzy z czego jedna to powrót, mamy aż 33,3% szans na to, że kafelek powróci na swoje miejsce. Przez te problemy niska ilość iteracji będzie prowadziła do natychmiastowego “ułożenia” puzzli i zakończenia gry.

Różna wielkość obrazków w galerii powodowała problemy. Niebyło widać niektórych części obrazka, a czasami obrazki były zbyt małe co było niezwykle irytujące w układaniu. Skalowanie do szerokości ekranu pomogło naprawić ten problem. Dzięki temu rozwiązaniu duże obrazki zawsze mieszczą się na ekranie, a małe obrazki poprawnie wypełniają ekran. Jedynym problemem wynikającym z tego rozwiązania jest rozpikselizownie małych obrazków, ale w ostatecznym rozrachunku ten sposób prezentowania obrazków daje więcej plusów niż minusów.