1. Tytuł projektu i autorzy projektu

20. Obrazkowe tetris

Autorzy projektu:

Ryszard Borzych
Sebastian Konik
Damian Koperstyński

2. Opis projektu

Celem projektu było stworzenie gry przypominającej Tetris. Zadaniem gracza jest ułożenie w jak najkrótszym czasie jednego ze zdefiniowanych obrazków poprzez prawidłowe umiejscowienie spadających z góry w losowej kolejności, kwadratowych klocków. Gracz ma możliwość obrotu fragmentów obrazka o 90° w prawą bądź lewą stronę. Ponadto użytkownik może wybrać ilość kawałków, na jaki podzielony zostanie obrazek oraz szybkość ich spadania.

3. Założenia wstępne przyjęte w realizacji projektu

Program powinien pozwalać na wybór jednego z kilku zdefiniowanych obrazków. Powinna również istnieć możliwość ustalenia na ile kwadratów obrazek ma zostać podzielony oraz tempa ich ruchu. Kwadraty mogą przesuwać się w dół skokowo o odległość równą bokowi kwadratu.

W wersji poszerzonej program powinien:

a) gwarantować w miarę płynny ruch kwadratów
b) kwadraty mogą być nie tylko odwrócone, ale również odbite symetrycznie wzdłuż pionowej lub poziomej osi
c) obraz wzorcowy mógłby być wyświetlany w studni w postaci czarno białej, a spadające kwadraty budowałyby go w wersji kolorowej

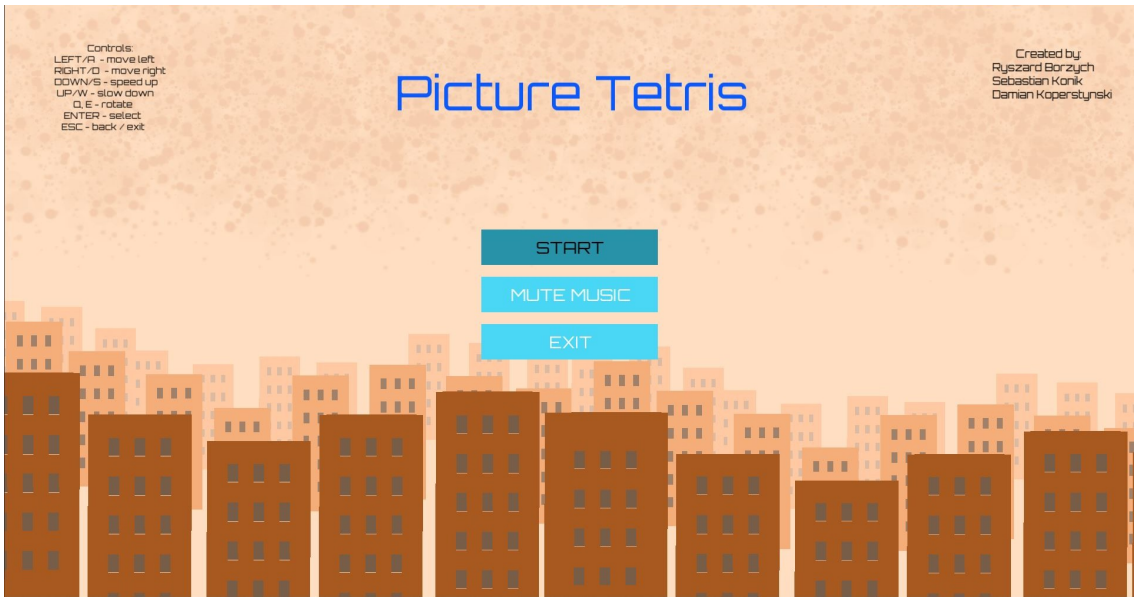
Udało nam się zrealizować wszystkie wymagania projektu, zarówno te podstawowe jak i rozszerzone z wyjątkiem podpunktu b) odpowiedzialnego za odbicie lustrzane. Powodem tego było zbyt duże zwiększenie poziomu trudności gry. Nie tylko nasza rodzina, ale także i znajomi nie radzili sobie z tym dodatkowym utrudnieniem, dlatego z obawy o monitory i klawiatury naszych znajomych usunęliśmy ten feature.

4. Analiza projektu

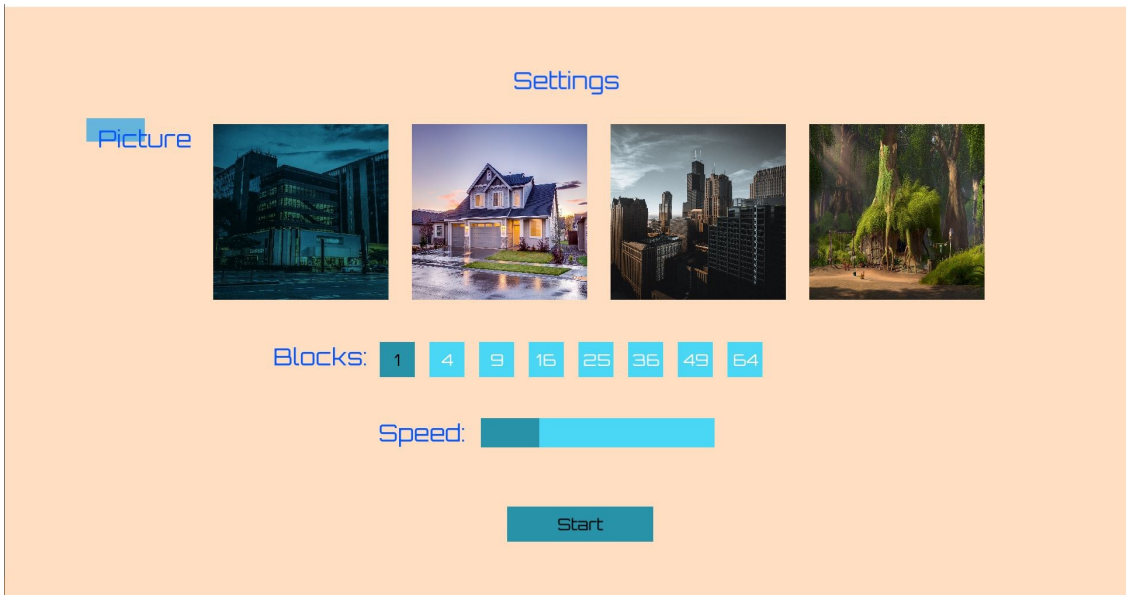
Danymi wejściowymi są 4 obrazki znajdujące się w folderze data/images, oraz dźwięki z gry. Dźwięki i obrazki można podmienić, przy zachowaniu takiego samego rozszerzenia i nazwy pliku. Oczekiwanym wyjściem był widok gry i rozłożenie obrazka na kawałki, aby moc go złożyć. Większa cześć projektu opierała się na widokach, zdefiniowaliśmy abstrakcyjny widok gry i wszystko działo się pomiędzy nimi: menu gry, ustawienia, sama rozgrywka, oraz widok końca gry. Zaimplementowaliśmy przyciski, oraz ich zestawy do wyboru odpowiedniego ustawienia. Zajęliśmy się także UI/UX, aby gracz wiedział, jakie ustawienie wybrał. Do ustawienia prędkości spadających klocków zdecydowaliśmy się do użycia slidera, z racji tego iż wartości liczbowa może być dla graczy zbyt abstrakcyjna.

Poniżej przedstawiono poszczególne elementy interfejsu użytkownika:

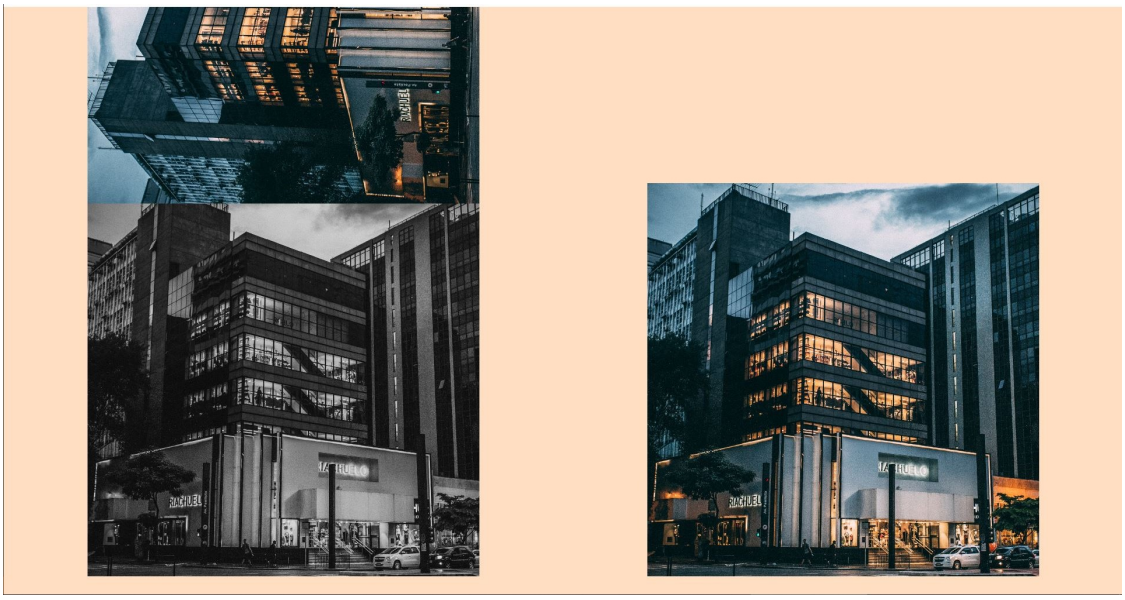
- Okno startowe



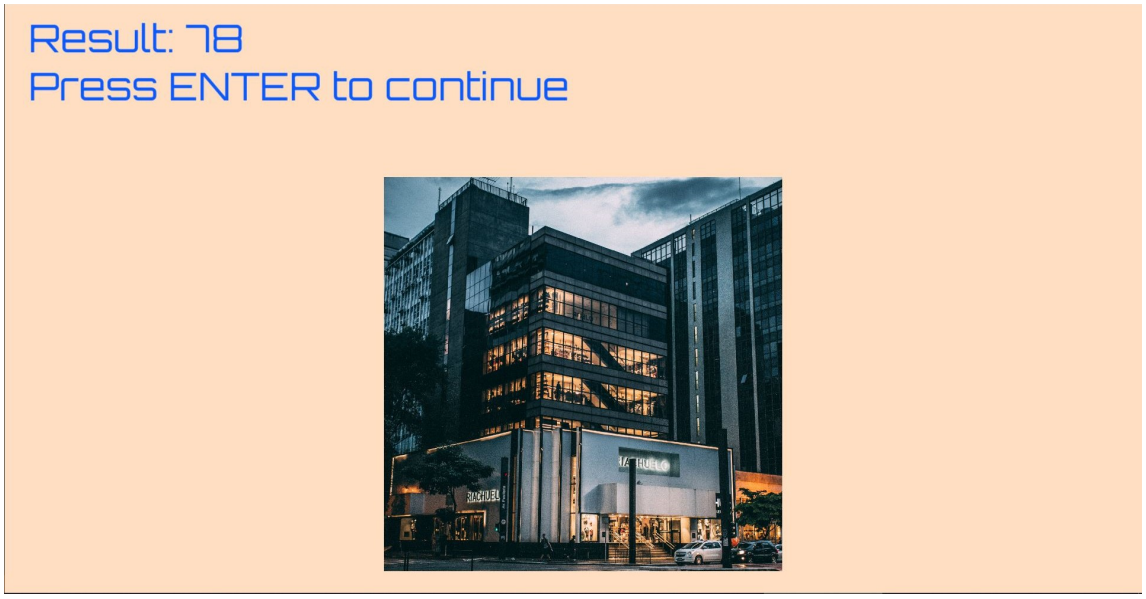
- Okno wyboru ustawień gry



- Ekran główny rozgrywki



- Ekran końcowy – po pomyślnym ułożeniu obrazka



5. Podział pracy i analiza czasowa

Podstawowe prace zostały podzielone na następujące segmenty:

- Dzielenie obrazków na fragmenty – 10 rg.
- Mechanika spadania – 10 rg.
- Stworzenie interfejsu użytkownika – 8 rg.
- Oprawa audio – 1 rg.
- Oprawa wizualna – 2 rg.
- Stworzenie komentarzy do kodu – 2 rg.

- Stworzenie dokumentacji – 5 rg.
- Optymalizacja – 6 rg.
- Koniec gry – 2 rg.

Prace nad projektem rozłożyliśmy równo pod względem trudności i pochłonięcia czasu:

- **Ryszard Borzych**:
  Oprawa graficzna
  Stworzenie interfejsu użytkownika
  Tworzenie dokumentacji
- **Sebastian Konik:**
  Mechanika spadania
  Optymalizacja
  Tworzenie komentarzy
  Tworzenie dokumentacji
- **Damian Koperstyński:**
  Dzielenie obrazków na fragmenty
  Oprawa audio
  Koniec gry
  Tworzenie dokumentacji

Prace nad projektem zaczęliśmy w pierwszej połowie maja. Całemu zespołowi praca zabrała około 60 roboczogodzin rozłożonych mniej więcej po 1/3 na każdego członka zespołu. Najwięcej czasu zajęło szlifowanie projektu i ustalenie szczegółów dotyczących realizacji projektu.

6. Opracowanie i opis niezbędnych algorytmów

Algorytm dzielenia obrazka na kawałki:

Przyjmuje długość boku wyrażoną w liczbie klocków. Obrazek zostaje podzielony poprzez funkcje SFML'a na szereg spritów, z których każdy reprezentuje jeden puzzle.

Sprity te są przechowywane w stosach przypisanych do danego rzędu.

Konwersja do skali szarości:

Kopia obrazek jest konwertowany na skalę szarości piksel po pikselu. Kolory dla każdego piksela zostają zwagowane z wagami: (0.12,0.72,0.07) i zwrócone, jako kolor o wszystkich parametrach o takiej samej wartości.

Gra właściwa (spadanie klocków):

Przy każdym odświeżeniu gry, jeżeli klocek nie jest położony zostaje on obniżony o wartość w zależności od prędkości. Gdy znajdzie się on na szczycie już ułożonych klocków w danej kolumnie zostaje sprawdzone czy jest on właściwie dopasowany pod względem rotacji i kolumny. Jeżeli te wartości się zgadzają to zostaje on umieszczony oraz zabrany z puli dostępnych jeszcze klocków.

Menu ustawień (interfejs):

Każda kontrolka ma stan aktywny lub nieaktywny. Wciśnięcie klawisza *enter* powoduje użycie przycisku ustawionego, jako aktywny. Menu po wciśnięciu klawisza przez gracza wysyła sygnał do odpowiednich przycisków przechowywanych w wektorze. Sama funkcjonalność danego przycisku nie jest przechowywana w nim samym, a w samym menu.

Koniec gry:

Gra dobiega końca, gdy liczba pozostałych klocków we wszystkich kolumnach wyniesie zero

Wynik końcowy:

Finalna wartość punktów zdobytych przez gracza jest wyliczana ze wzoru:

$$score = 100 * (speed + 1) * blocks - time$$

gdzie *speed* – szybkość spadania klocków wyrażona w procenta prędkości maksymalnej, *blocks* – długość boku kwadratu wyrażona w kawałkach obrazka, *time* – czas od rozpoczęcia wyrażony w sekundach.

Dokładna dokumentacja metod i klas projektu znajduje się w wygenerowanym przez doxygen dokumencie.

7. Kodowanie

Za narzędzie do stworzenia projektu wybraliśmy program **Visual Studio 2019** wraz z frameworkiem **SFML**. Projekt został napisany w języku **C++.** Nie wybraliśmy wxWidgets ze względu na konieczność operowania na obrazkach oraz małej liczbie wymaganych kontrolek oraz przez jego małą responsywność, co do wyglądu okna.

Pełna dokumentacja została wygenerowana przy pomocy **Doxygen'a**, a następnie przekonwertowana do **docs by doxygen.pdf** za pomocą oprogramowania **MikTex**.

Program składa się z głównej klasy, która odpowiada za wszystkie podstawowe procesy takie jak odświeżanie, odtwarzanie muzyki i utrzymywanie obecnego stanu aplikacji. Odpowiednie instancje programu (menu ustawień, gra, stan zakończenia) są dodawanego do niego. Oprócz tego program wykorzystuje następujące kontrolki:

- Button – przycisk (opcja), które jest rysowany wraz z zadanym tekstem
- Settings Button – podmenu przechowujące wiele przycisków (opcji) pozwalające na zwrócenie wybranej opcji
- Slider – szczególny rodzaj Settings Button, który zamiast w postaci serii przycisków jest przedstawiony jako jeden slider.
- Image Button – przycisk zamiast tekstu wyświetlający zadany obrazek

8. Testowanie

Testy odbywały się manualnie, z każda zmianą i wprowadzeniem nowej funkcjonalności, sprawdzaliśmy działanie aplikacji. Finalna wersja aplikacji przetestowana została przez niezależnych testerów (w tym babcię) w celu zlokalizowania bugów występujących w grze oraz ich naprawy.

9. Wdrożenie, raport i wnioski

Do skompilowania gry należy mieć zainstalowana bibliotekę SFML w wersji 2.5. Projekt wymagał od nas dokładnego zaprojektowania i przemyślenia całej architektury gry, natomiast manualne testy mimo tego iż były trochę czasochłonne, pomogły w dokładnych zrealizowaniu wymagań. W przyszłości można rozwinąć projekt dodając możliwość dodania większej ilości obrazków, nie tylko zmiany tych podstawowych, oraz wybór muzyki która będzie nam towarzyszyć podczas zabawy.