Programowanie Obiektowe 2			
Grupa	**Zespół**	**Temat Projektu**	**Rok Studiów**
2ID13A	Bartosz Kaczmarczyk Sebastian Kalemba	„Tetris w Bibliotece LibGDX”	Drugi

1. Wstęp Teoretyczny:

Niniejsze sprawozdanie winno potwierdzać postęp uczyniony przez nas w ramach pracy nad obowiązkowym projektem z przedmiotu „Programowanie Obiektowe 2”. Tematem projektu, którego zrealizowania podjęliśmy próbę był „Tetris” a językiem programistycznym „Java”.

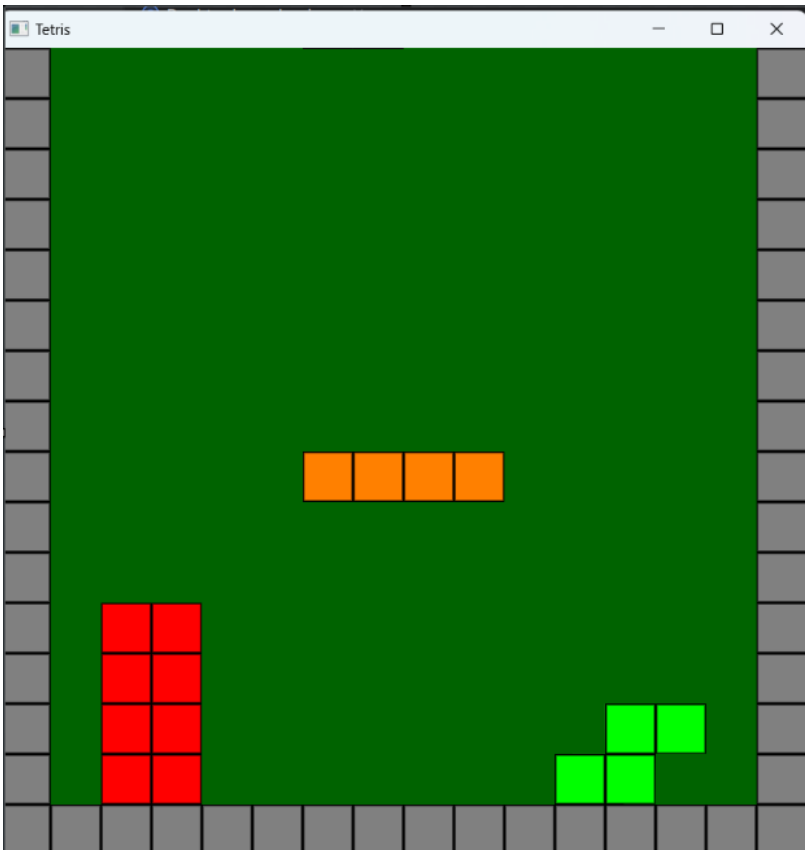
2. Opis:

2.1. Ogólny opis projektu

Celem naszego projektu było utworzenie interaktywnego, graficznego środowiska (gry) „Tetris”. Biblioteka z jakiej skorzystaliśmy to LibGDX. LibGDX to biblioteka programistyczna używana do tworzenia gier na różne platformy, takie jak Android, iOS, PC czy HTML5. Pomaga programistom w obszarach takich jak grafika, dźwięk, obsługa wejścia, fizyka i wiele innych, umożliwiając łatwe tworzenie wieloplatformowych gier przy użyciu języka Java.

2.2. Funkcjonalność projektu

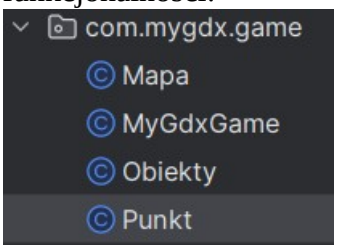
Jak sam tytuł wskazuje, rezultatem naszego projektu było utworzenie grywalnego prototypu popularnej gry wideo „Tetris”. Po uruchomieniu na ekranie wyświetla się prostokątne okienko, w jakim odgrywają się zdarzenia w obrębie funkcjonalności gry. Celem gracza jest układanie poziomych pasków z klocków. Jeżeli gracz (w jakiejkolwiek kombinacji) ułoży spójny poziomy pasek z klocków (od lewej do prawej), jest on usuwany a graczowi są dodawane punkty.



Okno gry

2.3. Obsługa projektu
Aby uruchomić projekt, należy plik build.gradle uruchomić za pomocą programu rozumiejącego język Java (preferowanym wyborem jest InteliJ). Następnie wybierając plik DesktopLauncher w dekstop/src/com.mygdx.game należy skompilować cały projekt. Uruchomi się wtedy wyżej wspomniane okno. Aby sterować, konieczne jest manewrowanie klawiszami w, s, d, które wpływają na to jak zachowują się poszczególne losowo generowane klocki. Jeżeli gracz doprowadzi do sytuacji w wyniku której nie jest możliwe wygenerowanie żadnego nowego klocka (brak miejsca), gra się kończy i zamyka.

2.4. Szczegółowy opis projektu
Oprócz klasy DesktopLauncher, projekt zawiera łącznie cztery inne klasy, które obsługują różne funkcjonalności:



Oto krótki opis co się dzieje w każdej klasie podczas kompilacji:
DesktopLauncher
>Importowane są potrzebne klasy z biblioteki LibGDX.
>Ustawiane są parametry konfiguracyjne dla aplikacji (np. szerokość i wysokość okna, liczba klatek na sekundę, tytuł okna).
>Tworzony jest obiekt konfiguracji (Lwjgl3ApplicationConfiguration) i ustawiane są na nim odpowiednie parametry.
>Inicjalizowane są pewne wartości związane z obiektem klasy Mapa.
>Tworzony jest obiekt Lwjgl3Application, który odpowiada za uruchomienie gry (MyGdxGame) z zadaną konfiguracją.
>Jest to główna klasa, a zatem wywoływane, tworzone i inicjalizowane są w tej klasie wszystkie inne klasy oraz obiekty.
Mapa
>Konstruktor Mapa(): Inicjalizuje różne parametry, w tym rozmiary planszy, tablicę Plansza_do_gry, oraz ustawia wartości domyślne. Następnie wykonuje test wyświetlający stan planszy.
>Metoda test(): Wyświetla aktualny stan planszy w konsoli.
>Metoda Kolizja(Obiekty obiekty, boolean poziomo, int odcinek): Sprawdza kolizję pomiędzy blokami na planszy a obiektami ruchomymi. Określa, czy jest możliwy ruch w danym kierunku.
>Metoda zapiszDoPliku(String nazwaPliku, int[][] tablica): Zapisuje zawartość planszy do pliku tekstowego.
>Metoda Rysuj_od_nowa(): Tworzy nową mapę w formie obrazu, zapisuje ją do pliku PNG, a następnie aktualizuje obiekt gry (MyGdxGame).
>Metoda Sprawdz_Czy_Jest_blok(int repeat): Sprawdza, czy na planszy istnieją pełne rzędy bloków, usuwa je i przesuwa pozostałe bloki w dół. Następnie aktualizuje planszę i obiekt gry.
MyGdxGame
>Metoda create(): Inicjalizuje muzykę, obiekt klasy Obiekty, teksturę tymczasową (tmp), oraz region tekstury (textureRegion). Tworzy również obiekt klasy Mapa.
>Metoda render(): Odpowiada za renderowanie gry. Sprawdza, czy wystąpił update planszy lub upadek bloku, a następnie rysuje planszę i aktualny klocek na ekranie. Wywołuje funkcję zapisującą stan planszy, jeśli flaga Zapis jest ustawiona na true.
>Metoda dispose(): Zwalnia zasoby, w tym obiekt klasy SpriteBatch.

>Metody update() i upadek(): Obsługują logikę ruchu klocka. update() reaguje na klawisze kierunkowe oraz spacje, a upadek() sprawdza, czy klocek powinien opaść o jedną pozycję w dół.

>Metoda zapis(): Tworzy zrzut ekranu i zapisuje go do pliku PNG o nazwie "tmp1.png". Aktualizuje teksturę tymczasową.
>Metoda jednorazowo(): Rysuje początkową planszę.

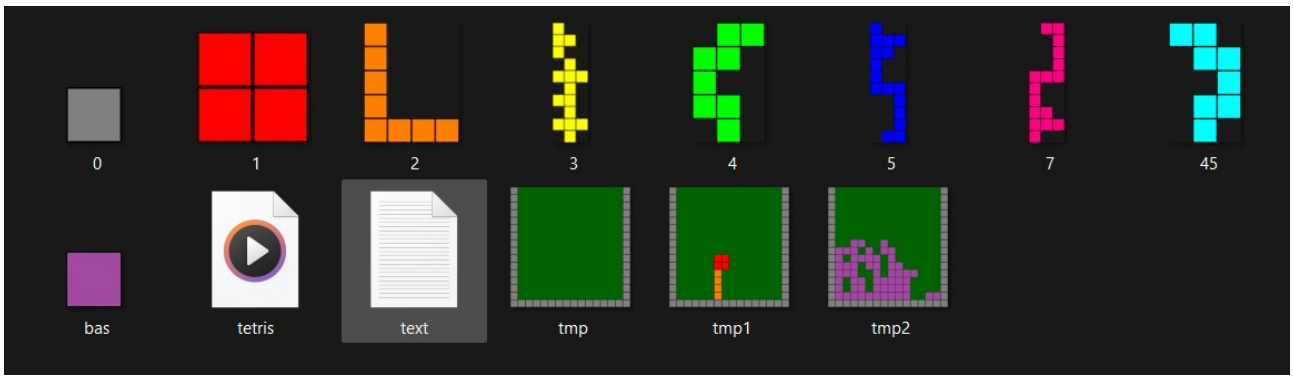
Obiekty
Pola klasy:
x i y: Określają pozycję obiektu na planszy.
tekstura: Przechowuje teksturę obiektu.
blok: Lista punktów (klasa Punkt) reprezentujących bloki składające się na obiekt.
szerokosc i wysokosc: Określają wymiary obiektu.
X_Region i Y_Region: Określają fragment tekstury, który ma zostać wyświetlony.
ktory_klocek i ktory_obrot: Przechowują informacje o aktualnie wybranym klocku i jego obrocie.
max_y: Przechowuje maksymalną wartość na osi Y dla obiektu.
>Metoda Losowanie():
Inicjalizuje losowy obiekt (klocek) poprzez wybór liczby od 1 do 5. Tworzy teksturę obiektu na podstawie wylosowanej liczby. Dodaje punkty (bloki) do listy blok w zależności od wylosowanego klocka.
>Metoda Tranformacja():
Modyfikuje obiekt w zależności od jego typu:
>Metoda Poruszanie(int odcinek, boolean Poziomo):
Zmienia pozycję obiektu o określoną wartość odcinek, zarówno w poziomie, jak i w pionie, w zależności od wartości logicznej Poziomo.
>Metoda Zapisz_do_tablicy_polozenie(int tab[][]):
Zapisuje aktualne położenie obiektu do tablicy tab, oznaczając bloki na planszy wartością 7.

Punkt
Pola klasy:
x i y: Przechowują współrzędne punktu.
Konstruktor Punkt(int x, int y):
Inicjalizuje obiekt klasy Punkt przyjmując współrzędne punktu jako argumenty.

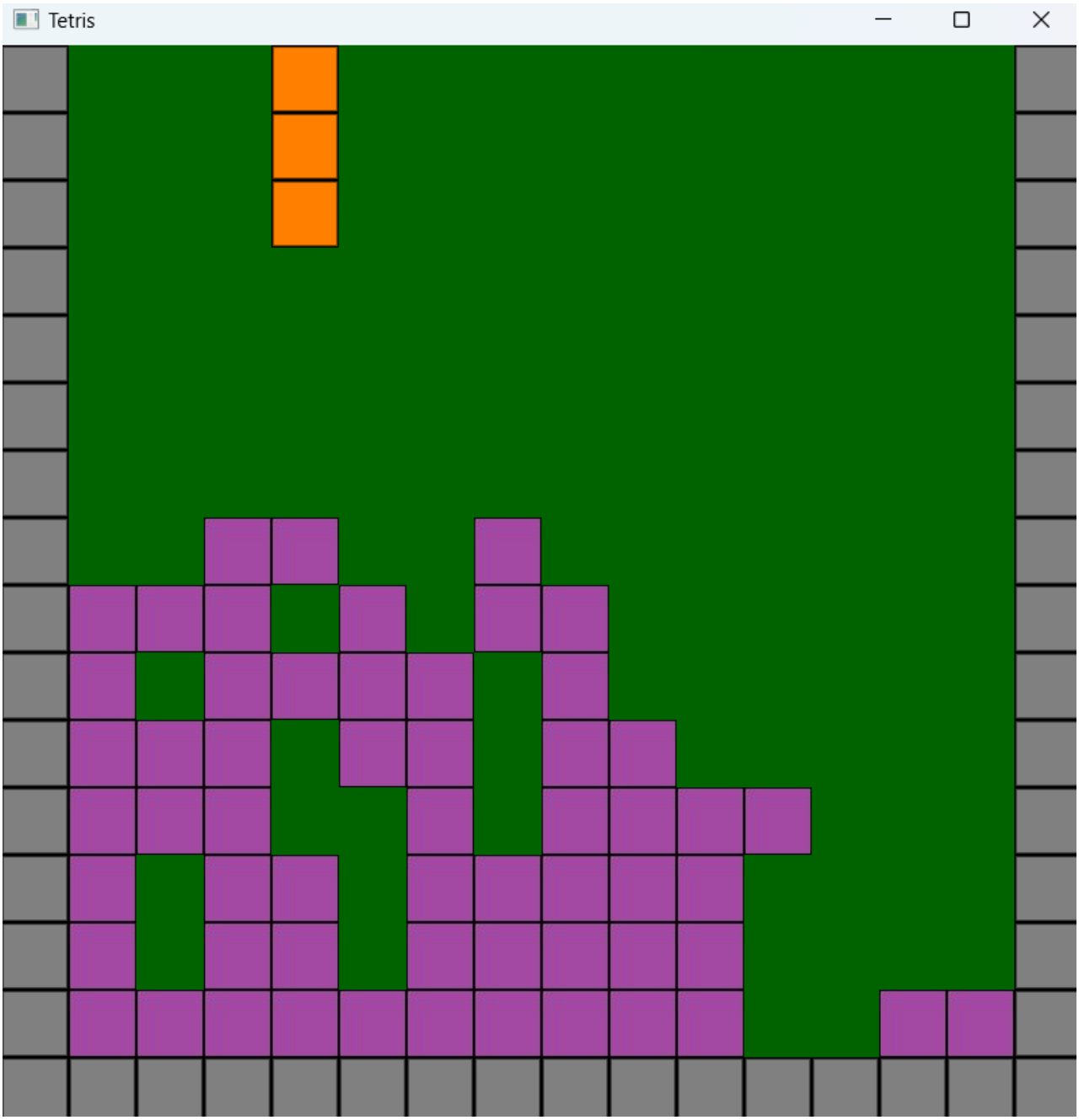
Na ekranie wyświetlane są bloki, z każdy z nich posiadający swoje unikalne współrzędne na bitmapie. Punkty te stanowią zarówno miejsce kolizji, jak i potencjalne miejsce, gdzie może dotrzeć nasz bloczek. Kiedy bloczek osiągnie dane współrzędne, zaznaczone na macierzy kolizji (plik text.txt), zniknie, a inne bloki zamieniają się w ten sam kształt. Następnie, w celu imitacji opadania, pozostałe bloki przemieszczają się o trzy jednostki w dół.

Wartym wspomnienia faktem jest to, że napotkaliśmy wiele problemów w trakcie pracy nad projektem. Przede wszystkim wystąpiły problemy z implementacją oraz obsługą oprawy graficznej, zarówno logiczne jak i programistyczne. Błędy z kolizją oraz błędy z wychodzeniem poza indeks tablicy również utrudniały nam prace nad projektem. Właśnie ta trójka – graficzne, indeksowe oraz kolizyjne były problemami jakie zabrały nam najwięcej czasu na debuggowaniu.

Na potrzeby gry utworzyliśmy wiele plików graficznych oraz przygotowaliśmy plik dźwiękowy w formacie .wav, jaki stanowi ścieżkę dźwiękową gry, która jest stale odtwarzana w tle.



Pliki zasobów



Rezultat po utworzeniu poziomego paska

2.5. Podział prac

Sebastian Kalemba:
>Znajdywanie i weryfikowanie informacji na temat biblioteki LibGDX
>Projektowanie oprawy graficznej
>Implementacja oprawy graficznej
>Dokonanie sprawozdania
>Dokonanie dokumentacji

Bartosz Kaczmarczyk:
>Zaplanowanie, zaprojektowanie i zaimplementowanie kodu źródłowego gry
>Implementacja zasad gry
>Debugowanie
>Testowanie
>Archiwizacja projektu

3. Wnioski

Dokonawszy zadań, które otrzymaliśmy w ramach realizacji projektu z przedmiotu „Programowanie Obiektowe 2”, spełniliśmy wymagania, które postawiliśmy sobie w ramach tematu „Tetris”. Zrozumieliśmy od bardziej technicznej strony jak należy implementować rozwiązania graficzne oraz audio-wizualne powiązane z językiem Java, a także nabyliśmy doświadczenie jakie może okazać się konieczne w przyszłych projektach tego typu. Tym samym, uznajemy nasz projekt za ukończony a jego realizację za wypełnioną sukcesem.